NAVAL POSTGRADUATE SCHOOL
Monterey, California

DTIC
SELECTED
MAR 2 4 1995
G

THESIS

A WHOLESALE LEVEL CONSUMABLE ITEM INVENTORY MODEL FOR NON-STATIONARY DEMAND PATTERNS

by

Glenn C. Robillard

March 1994

Thesis Advisors: Thomas P. Moore
Alan W. McMasters

Approved for public release; distribution is unlimited.

19950322 076

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	March 1994	Master's Thesis

4. TITLE AND SUBTITLE **A WHOLESALE LEVEL CONSUMABLE ITEM INVENTORY MODEL FOR NON-STATIONARY DEMAND PATTERNS**

5. FUNDING NUMBERS

6. AUTHOR(S) Glenn C. Robillard

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(maximum 200 words)*

The U.S. military presently manages about 88 billion dollars in spare and repair parts, consumables, and other support items. Department of Defense (DOD) inventory models which help wholesale item managers make inventory decisions concerning these items are based on the assumption that mean demand remains constant over time. In DOD this assumption is rarely met. During periods of declining demand, such as that associated with force reduction or equipment retirement, the inventory models usually keep stock levels too high, generating excess material. Recently, the amount of excess in DOD was estimated to be as high as 40 billion dollars. On the other extreme, during periods of increasing demand, the models generally provide too little stock, resulting in poor weapons system support. The purpose of this research was to develop an inventory model which does not rely on the assumption that mean demand is stationary. Use of the model would be appropriate when a known or predictable increase or decrease in mean demand is forecasted. Through simulation the model's performance was evaluated and compared with that of the Navy's Uniform Inventory Control Program (UICP) model. The results indicate that the proposed model significantly outperforms the existing model when mean demand is non-stationary. Additionally, the results indicate that the proposed model's performance is equal to or better than the existing Navy model under many stationary mean demand scenarios.

14. SUBJECT TERMS Inventory Management, Inventory Models, Non-Stationary Demand, Declining Demand, Inventory Simulation, Consumable Items

15. NUMBER OF PAGES 273

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

Approved for public release; distribution is unlimited.

A Wholesale Level Consumable Item Inventory Model
for Non-Stationary Demand Patterns

by

Glenn C. Robillard
Lieutenant, Supply Corps, United States Navy
B.S., University of Massachusetts, 1978
M.Ed., University of Massachusetts, 1982

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
March 1994

Author: Glenn C. Robillard

Glenn C. Robillard

Approved by: Thomas P. Moore

Thomas P. Moore, Thesis Advisor

Alan W. McMasters

Alan W. McMasters, Thesis Co-Advisor

Gordon H. Bradley, Second Reader

Peter Purdue, Chairman
Department of Operations Research

ABSTRACT

The U.S. military presently manages about 88 billion dollars in spare and repair parts, consumables, and other support items. Department of Defense (DOD) inventory models which help wholesale item managers make inventory decisions concerning these items are based on the assumption that mean demand remains constant over time. In DOD this assumption is rarely met. During periods of declining demand, such as that associated with force reduction or equipment retirement, the inventory models usually keep stock levels too high, generating excess material. Recently, the amount of excess in DOD was estimated to be as high as 40 billion dollars. On the other extreme, during periods of increasing demand, the models generally provide too little stock, resulting in poor weapons system support. The purpose of this research was to develop an inventory model which does not rely on the assumption that mean demand is stationary. Use of the model would be appropriate when a known or predictable increase or decrease in mean demand is forecasted. Through simulation the model's performance was evaluated and compared with that of the Navy's Uniform Inventory Control Program (UICP) model. The results indicate that the proposed model significantly outperforms the existing model when mean demand is non-stationary. Additionally, the results indicate that the proposed model's performance is equal to or better than the existing Navy model under many stationary mean demand scenarios.

THESIS DISCLAIMER

The reader is cautioned that computer programs developed in this research may not have been exercised for all cases of interest. While every effort has been made, within the time available, to ensure that the programs are free of computational and logic errors, they cannot be considered validated. Any application of these programs without additional verification is at the risk of the user.

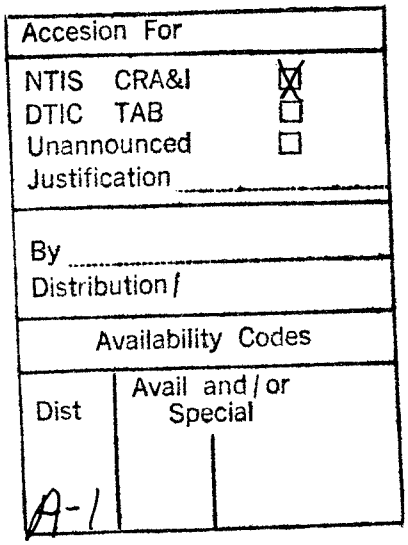


TABLE OF CONTENTS

I. INTRODUCTION 1

A. OVERVIEW 1

B. OBJECTIVES AND SCOPE OF THE RESEARCH 2

C. LITERATURE REVIEW 4

D. ORGANIZATION OF THE THESIS 9

II. THE UICP INVENTORY MODEL 10

A. INTRODUCTION 10

B. BASIC ASSUMPTIONS AND DEFINITIONS 11

C. INVENTORY MODEL 13

D. FORECASTING QUARTERLY DEMAND IN UICP 16

III. THE MODIFIED SILVER MODEL 18

A. INTRODUCTION 18

B. ASSUMPTIONS AND DEFINITIONS 19

C. MODEL DEVELOPMENT 22

1. Determination of the Reorder Point 22

2. Determination of the Order Interval (T) 23

3. Determination of the Order Quantity 24

D. PARAMETERS AND IMPLEMENTATION 27

1. Standard Deviation of Forecast Errors 27

2. Estimating Demand Variability 33

3. Normality Assumption for Forecast Errors 34

4. Implementing Periodic Reviews 37

5. Estimating the Order Interval Length (T) 41

IV. SIMULATION 43

A. INTRODUCTION 43

B. ASSUMPTIONS 44

C. SIMULATION STRUCTURE AND DESIGN 45

1. Demand Observations 47

2. Forecasting 50

3. Levels Setting 51

4. Supply Demand Reviews 52

D. INITIALIZATION AND TERMINATION 55

1. Number of Replications 55

2. Seed Selection 57

3. System Parameters 59

4. Initial Conditions 60

5. Terminating Conditions 62

V. RESEARCH METHODOLOGY AND RESULTS 64

A. OVERVIEW 64

B. EXPERIMENTAL FACTORS 65

1. Demand Profile 65

2. Demand and Lead Time Distribution 66

3. Unit Price 67

4. Other Parameters 67

a. Risk 67

b. Buffer 68

c. Maximum Order Cycle Length During Decline 71

C. SIMULATION RESULTS 72

1. Stationary Demand 72

2. Cyclic Demand 78

3. Declining Demand 81

4. Increasing Demand 86

D. DECISION ANALYSIS 90

VI. SUMMARY, CONCLUSIONS, AND RECOMMENDATIONS 98

A. SUMMARY 98

B. CONCLUSIONS 99

C. RECOMMENDATIONS 100

APPENDIX A. SIMULATION CODE 102

APPENDIX B. WELCH GRAPHS 212

APPENDIX C. SCENARIO LISTING 214

APPENDIX D. DEMAND PROFILE GRAPHS 218

APPENDIX E. EXPERIMENTAL RESULTS 221

APPENDIX F. STATIONARY DEMAND GRAPHS 250

APPENDIX G. CYCLIC DEMAND GRAPHS 251

APPENDIX H. DECLINING DEMAND (STEEP TREND) GRAPHS 252

APPENDIX I. DECLINING DEMAND (SLOW TREND) GRAPHS 253

APPENDIX J. INCREASING DEMAND (STEEP TREND) GRAPHS 254

APPENDIX K. INCREASING DEMAND (SLOW TREND) GRAPHS 255

LIST OF REFERENCES 256

INITIAL DISTRIBUTION LIST 259

EXECUTIVE SUMMARY

In the past decade the Department of Defense's (DOD) inventory management of secondary items has come under intense Congressional scrutiny. Secondary items are consumable and repairable spare and repair parts and other support items that are needed to maintain the readiness of the military forces' weapon systems and support military personnel. The total value of this material was $88.1 billion in fiscal year 1991. The vast majority ($73 billion) of these items were stocked as demand-based items.

Inventory models which help DOD wholesale item managers make inventory decisions concerning these items are based on the assumption that mean demand remains constant over time. Since, in reality, mean values for demand change over time, the existing models cannot directly compensate for these changes. During periods of declining demand, such as that associated with force reduction or equipment retirement, the inventory models usually keep stock levels too high, generating excess material. Recently, the amount of excess in DOD was estimated to be as high as $40 billion. On the other extreme, during periods of increasing demand, the models generally provide too little stock, resulting in exceptionally poor levels of customer service.

In this study an alternative replenishment strategy is proposed which does not rely on the assumption that mean demand is stationary. The basis for the research is a model developed by E.A. Silver for probabilistic demand with a time varying mean. Silver's model is a three-stage procedure, determining when to order, the number of

periods to cover, and the order quantity. Although the model assumes demand is probabilistic, the determination of the length of the order cycle is based upon the deterministic Silver-Meal heuristic, selecting the order quantity so that total relevant costs are minimized over the period that the replenishment quantity will support. Silver's model is extended and modified as necessary to work within existing DOD inventory information systems, to comply with DOD mandated constraints, and to handle the uncertainty of a non-deterministic replenishment lead time. The resulting model is called the modified Silver model. Since the model requires the ability to forecast over a specified time horizon, use of the model would be appropriate when a known increase or decrease in mean demand is forecasted.

Evaluation of the modified Silver model is based on a Monte Carlo simulation. The baseline measurement is the performance of the current Navy Uniform Inventory Control Program (UICP) model for consumable items under the same simulated demand scenarios. Since the Navy model is based on the general DOD model, the results should have direct applicability to other DOD components' (DLA, Army, Air Force) models. Both simulations approximate the inventory management of a single consumable item for as many as 120 quarters. The simulation experiments include a variety of run characteristics, system parameter settings, and generated demand profiles.

Testing of the modified Silver model using a stationary mean demand forecast demonstrates comparable or slightly improved performance over the UICP model. This supports the assertion that the models are nearly equivalent under the assumption

that mean demand is stationary. Testing of the models when mean demand is varying and estimates of the varying mean are included in the forecast clearly demonstrates that the modified Silver model outperforms the existing UICP model. In declining demand scenarios, the modified Silver model significantly reduces both excess inventory at the end of the simulation interval and the total cost over the simulation interval, with no reduction in average customer wait time. In increasing demand scenarios, the model significantly reduces average customer wait time at an overall lower total cost.

I. INTRODUCTION

A. OVERVIEW

In the past decade the Department of Defense's (DOD) inventory management of secondary items has come under intense Congressional scrutiny. Secondary items are consumable and repairable spare and repair parts and other support items that are needed to maintain the readiness of the military forces' weapon systems and support military personnel. The total value of this material was $88.1 billion in fiscal year 1991. The vast majority ($73 billion) of these items were stocked as demand-based items [Ref. 1]. According to one General Accounting Office (GAO) report, DOD has "wasted billions of dollars on excess supplies, burdened itself with the need to maintain them, and failed to acquire the tools or expertise to manage them effectively" [Ref. 2]. The same report estimated DOD's excess inventory at about $40 billion.

Within DOD, inventory control points (ICP's) have the primary responsibility for the wholesale management of secondary items. The DOD model used for demand-based requirements determination is an (s,S) inventory model. In determining requirements three factors are considered - safety level, lead time requirements, and the economic order quantity (EOQ). Safety levels and lead time requirements are combined to determine the reorder level (s). If the inventory position (on-hand plus on-order minus back-ordered stock) falls below the reorder level, then a replenishment is made to bring the inventory position up to the order-up-to-level (S). The order-up-

to-level is primarily a function of the economic order quantity. A fundamental assumption of the reorder point and economic order quantity formulas used by DOD is that the demand rate is stationary over time. Unfortunately in most environments, including DOD, this assumption is rarely met. Continued use of this assumption has been linked by GAO to the buildup of excess inventories in the military supply system[1] [Ref. 3]. GAO's findings include the recommendation for DOD to adopt a replenishment strategy that can be used when the mean demand rate is non-stationary over time. With the current and planned reduction in the size of the Department of Defense, and the ensuing decline in secondary item demand, the need for such a model has never been more pressing.

B. OBJECTIVES AND SCOPE OF THE RESEARCH

The primary objective of this research is to develop and test a model based on economic order quantity principles for probabilistic time-varying mean demand with stochastic lead times. The model will be used to determine both a reorder point and a reorder quantity. In designing the model, the following elements are considered crucial:

- The model should significantly reduce excess inventories yet still maintain adequate levels of customer service.

---

[1]For a discussion of excess inventory in the Navy supply system, see Lilli and Husson's thesis [Ref.4]. For a discussion of other contributing causes see Perry's study [Ref. 5] or the GAO report on the shortcomings in requirements determination processes in the DOD [Ref. 6].

- The model should be designed to fit into existing inventory information systems, such as the Navy's Uniform Automated Data Processing System for Inventory Control Points (UICP), with minimal changes required in software and no changes in hardware.

- The model should be tested over a wide range of scenarios, representing increasing and declining demand, as well as a full range of demand variability.

- The model should be simple to understand.

- The model should meet all DOD mandated constraints.

The basis for the research is a model developed by E.A. Silver for probabilistic demand with a time varying mean [Ref. 7]. The model is extended and modified as necessary to meet the above requirements. The resulting model is called the "modified Silver model."

Evaluation of the new model is based on a Monte Carlo simulation. The baseline measurement is the performance of the current Navy UICP model for consumable items under the same simulated demand scenarios. Since the Navy model is based on the general DOD model, the results should have direct applicability to other DOD component (DLA, Army, Air Force) models.

The model has particular application to items experiencing a known or predictable decline or increase in mean demand, such as with system retirement, planned reductions or increases in component population, or engineering design changes of the weapons system component that contains the item. Since all changes in mean demand are not predictable, the effects of using the model when mean demand is stationary is also studied.

C. LITERATURE REVIEW

There has been a considerable amount of development of inventory models when mean demand is assumed to be stationary. There has also been a considerable amount of attention paid to the deterministic version of the non-stationary demand problem. However, the number of papers devoted to probabilistic, non-stationary demand is very limited. Since we are concerned exclusively with non-stationary demand, several of the more important papers for both the deterministic and the probabilistic case for non-stationary demand are discussed in the following paragraphs.

The most often discussed procedure for the time-varying, deterministic case is the Wagner-Whitin (W-W) dynamic lot-sizing algorithm [Ref. 8]. Although the procedure provides a true optimal solution using a dynamic programming approach, it has often been ignored for practical use because of the amount of computational effort required and the possible need for a well defined ending point for the demand pattern [Ref. 9: p. 231]. Consequently many heuristics have been developed which are both easier to implement and computationally less demanding.

Of the several heuristic approaches, the Silver-Meal (S-M) heuristic [Ref. 9] has received a significant amount of review. The heuristic is a simple modification of the basic economic order quantity model for the discrete case. The strategy of the heuristic is to select the order quantity so that the total relevant costs (ordering and holding) are minimized for the time the replenishment quantity will last. In performance tests of the S-M heuristic, the W-W algorithm, and the EOQ, the average

cost penalty for using the heuristic over the algorithm is less than 1%. Furthermore, whenever the W-W algorithm significantly outperformed EOQ, so did the S-M heuristic [Ref. 9: p. 237]. Blackburn and Millen [Ref. 10] also showed that in a rolling horizon implementation where a firm has limited information about the future, the S-M heuristic is superior to the W-W algorithm in terms of cost effectiveness.

Ritchie and Tsado [Ref. 11] compared the S-M heuristic, among others, to a marginal cost approach. Using the marginal cost approach, the economic order quantity is determined by increasing the lot size as long as the marginal savings in ordering costs are greater than the marginal cost increase in inventory holding costs. In the case of a life-cycle demand pattern, tests indicated that the marginal cost approach performed better than the S-M heuristic. The life cycle demand patterns included a period of increasing demand followed by a stationary period followed by a decreasing period. All comparisons were made to baseline W-W optimal values.

Cline, Foote and Schlegel [Ref. 12] compared the W-W algorithm, S-M heuristic, along with the EOQ and several other less common heuristics, for a single-stage lot-size production problem with probabilistic demand. They concluded that the W-W algorithm clearly worked best if minimizing shortages is the criterion of choice. Otherwise, the S-M heuristic or EOQ were among the equally good choices.

Several other papers in the literature deal with specific patterns of time varying deterministic demand. Donaldson [Ref. 13] determines an optimal strategy for linear increasing demand. Ritchie and Tsado [Ref. 14] show that using the EOQ in cases of linear increasing demand results in only a small cost penalty when compared with

Donaldson's optimal value. In either case, because of their limited application these works provide little insight into the general case of problems associated with this study.

As indicated earlier, the amount of research dealing with non-stationary probabilistic demand is much more limited. The inclusion of uncertainty in demand alone can significantly complicate the problem from a conceptual viewpoint. Having the additional uncertainty in replenishment lead time, as well as allowing for a time varying mean demand only compounds this already complex problem.

In a 1978 paper, Silver [Ref. 7] provides a relatively simple approach to the probabilistic lot-sizing problem, using in part, a deterministic technique to determine the length of an order cycle. His model forms the basis of this research. Silver's model is a three stage approach - deciding when to order, the number of periods the order should cover, and the order quantity. The model assumes that replenishment lead time is fixed. Silver's model is discussed in great detail in Chapter III of this thesis.

Askin [Ref. 15] develops a similar, although somewhat more complex procedure, where the probabilistic nature of demand is included in determining the length of the order cycle. Using his approach, the length of the order cycle is determined by finding the cycle length T that minimizes expected cost over the forecast horizon. Again, replenishment lead time is assumed to be fixed. One important assumption of his basic model is that once an order is placed for T time periods, another order is not placed for another T periods. Askin offers a

modification to the model for an every-period-review approach but the model becomes significantly more complex. In either case the probability distribution of demand must be known.

More recently a near-myopic heuristic is provided by Bollapragda and Morton [Ref. 16]. Myopic policies order as if left over inventory from the current order could be salvaged at full value, allowing the problem to be solved easily without extensive knowledge of the future or dynamic programming. In this case "extra" units ordered during one replenishment cycle, if not used, could simply be applied to offset the next period requirements. This could lead to serious problems in the case of declining demand where there may be insufficient requirements to be offset. In such cases they hypothesize the heuristic is near myopic. Their heuristic involves first solving the optimal problem for a series of stationary demand problems and tabulating the (s,S) results. The non-stationary problem is then approximated by the stationary problem. This is done by averaging the demand parameters over an estimate of the replenishment time and reading the corresponding (s,S) values from the stationary tables. Bollapragda and Morton compare their "newsboy" heuristic and Askin's procedure to a dynamic programming solution. Overall, the heuristic averaged 1.7% error as compared to Askin's procedure, which averaged 2.0% error. The results were presented for relatively low varying demand only (σ/μ=0.1 and 0.3).

Another approach to a trended economic order quantity is currently under investigation for the Defense Logistics Agency (DLA) by analyst at the Defense Electronic Supply Center [Ref. 17]. Although published results are unavailable,

research as been directed towards increasing or declining demand with either exponential or linear trends. Based on the limited information available, a variable safety level model is being developed where mean demand is assumed to follow one of these preset curves. Curve parameters are obtained from a regression model. The model then solves iteratively for a deterministic EOQ and a variable safety level.

Lilli and Husson [Ref. 4] specifically addressed the issue of declining demand and the generation of excess assets. Using simulation, they first show that improved forecasting alone will not completely solve the problem of excess assets following a declining demand period. To eliminate excess inventories, they develop a model which reduces both the order quantity and reorder level proportionally to the population decline over the length of the declining cycle. The technique was successful in reducing excess inventory, however, the improvement comes at the expense of customer service.

Many of the theoretical papers discussed in the previous paragraphs provide insight in dealing with changes in demand. The most important insight is the complexity of the problem when demand is probabilistic. Silver's work appears to provide the most straightforward and general application. Askin's improves on Silver's model by including the probabilistic nature of demand in the determination the length of the order cycle. The model is more complex though and requires explicit knowledge of the demand distribution. Bollapragda and Morton provide a new approach, although it also requires knowing the probability distribution of demand.

With the large scale inventory maintained by DOD, this requirement creates serious drawbacks for either of the latter models.

D. ORGANIZATION OF THE THESIS

This thesis consist of six chapters. Chapter II provides an overview of the current UICP inventory model for consumable items. Chapter III provides a detailed description of the modified Silver model. The first few sections are devoted to the introduction of notation and model assumptions. The remainder of the chapter describes the basic Silver model, extensions and modifications to the model, and the estimation of model parameters. Chapter IV provides an overview of the simulation software, followed by a detailed description of the simulation structure and implementation assumptions. Chapter V provides the experimental design used to compare model performance and presents the results of the simulation experiments. Chapter VI presents the summary, conclusions, and recommendations of this research.

II. THE UICP INVENTORY MODEL

A. INTRODUCTION

The principal policy for the Department of Defense (DOD) concerning procurement cycles and safety levels of supply for secondary consumable items is provided in DOD Instruction 4140.39 [Ref. 18]. The objective of the policy guidelines stated in this instruction is:

> To minimize the total of variable order and holding costs subject to a constraint on time weight-weighted, essentiality-weighted requisitions short.

The mathematical model specified in this instruction parallels the lot size reorder point model for the backorders case as described by Hadley and Whitin [Ref. 19: p. 181-195]. The model assumes demand is stochastic, and that the mean rate of demand remains constant over time.

OPNAV Instruction 4440.23 [Ref. 20] further specifies policy within the Navy. Partial documentation for the Navy specific inventory model, including computational methods and constraints, can be found in NAVSUP Publication 553 [Ref. 21]. The current system design and the specifications for the computer program that implements the model are described in Functional Description (FD) PD-82 published by the SPCC, Code 046 [Ref. 22]. The purpose of this chapter is not to reiterate these documents, but simply to provide the reader with enough detail to have a general understanding of the UICP consumable item inventory model.

B. BASIC ASSUMPTIONS AND DEFINITIONS

In order to understand the inventory model currently employed by SPCC for consumable items, some basic assumptions and definitions are necessary. Replenishment decisions are based on the decision variables, reorder quantity R and order quantity Q, and inventory position (IP) which is defined to be the quantity on hand plus on order minus the quantity backordered. With the continuous-review (Q,R) policy used by the Navy, when the IP reaches or falls below the reorder quantity R, the order quantity Q plus IP-R units are ordered for stock replenishment.

Each time an order is placed, certain setup or administrative ordering costs are incurred (denoted here by A). DOD Instruction 4140.39 provides detailed guidance about the types of costs associated with ordering an item for inventory. These costs are divided into two categories, fixed and variable. Only variable costs, those that will vary as a function of the number of times an order is placed, are to be included in the determination of A. SPCC assigns administrative ordering cost based on the type of item, procurement method, item mark code, and the dollar value of the order. The item mark code is a categorization code based on forecasted quarterly demand and unit price.

When material is physically held in stock certain variable holding costs are incurred. The variable holding cost rate, denoted here by I, includes the costs associated with capital, obsolescence, and storage. This rate is often expressed as a fraction or percentage of unit cost per year; i.e., the cost to hold one dollar's worth of material in inventory for one year. For consumable items, SPCC currently uses 0.23

for the value of I, which consists of 0.10 for capital (time preference), 0.12 for obsolescence[2], and 0.01 for storage costs.

Shortages occur when there is insufficient stock on-hand to fill a requisition, resulting in a "backordered" requisition. Associated with each shortage are costs, which may be time-weighted or independent of time. Time-weighted costs in the military system are those costs which increase with the length of time the shortage lasts. Time independent costs are incurred just once at the start of the shortage. An example of such a cost is notifying the customer that the part is not in stock. In the model used by SPCC the shortage cost is computed using the time-weighted method, thus the units on the shortage cost rate (λ) are dollars per requisition year short; i.e., λ is the cost of being short one requisition for one year. In reality, λ is an implied shortage cost based on a specified service level and the available budget. The true cost of a shortage is unknown.

Essentiality (E) is the relative importance of a given item in an inventory to the military readiness of the weapon system of which it is a component. At SPCC this value is assumed to have a uniform value of 0.5 for all items and thus can be ignored. The model therefore uses a shortage cost to reflect, in some sense, a measurement of an item's military essentiality.

---

[2] This value is actually composed of 0.10 for obsolescence and 0.02 for pilferage and inventory adjustments.

C. INVENTORY MODEL

The constrained optimization problem of minimizing total ordering and variable holding costs subject to a constraint on time-weighted, essentiality-weighted requisitions short can be written in a general, unconstrained form (the Lagrangian function) as:

Find non-negative Q and R which minimize

$$TVC(Q,R) = \frac{4AD}{Q} + IC\left[R + \frac{Q}{2} - \mu + B(Q,R)\right] + \frac{B(Q,R)\,\lambda E}{S}$$

where:

Q is the reorder quantity;
R is the reorder point;
A is the administrative ordering cost;
I is the holding cost rate;
λ is the shortage cost;
E is the item essentiality;
D is mean demand in units per quarter;
C is unit cost;
μ is mean lead time demand;
B(Q,R) is the expected number of backorders (a function of Q and R);
S is the expected number of units per requisition.

The first term in this "cost equation" represents the average annual ordering costs for the item. The second term represents the average annual holding costs under long run steady state conditions. The final term in the equation represents the essentiality-weighted average annual number of requisition-years of shortages multiplied by the shortage cost rate λ. The shortage cost rate (λ) is, in reality, a Lagrange multiplier. Because all real world factors cannot be integrated easily into

the total variable cost equation, the Navy imposes further constraints on the solutions to this equation.

To determine an initial value for the reorder point R, the Navy model first makes use of the optimality condition obtained by taking the partial derivative of TVC(Q,R) with respect to R and setting the result equal to zero. Unfortunately, the resulting expression contains an implicit function of Q and R, making it difficult to solve the optimality condition explicitly for R. Instead, SPCC uses an approximation technique to determine the value of R. The resulting optimality condition is simplified to finding the smallest R such that:

$$1-F(R) = \frac{SIC}{SIC+\lambda E}$$

where F(R) is the cumulative distribution function describing the probability that the random variable representing lead time demand will be less than or equal to R. The right hand expression is defined to be optimal risk. Risk is defined as the probability of being out of stock during a procurement lead time (L). Substituting average requisition frequency (S=D/W) into the above expression yields:

$$1-F(R) = \frac{DIC}{DIC+\lambda WE} = P$$

where P represents the unconstrained stock-out risk at (unconstrained) optimality. This is the initial risk equation used at SPCC. Prior to determining the initial value of R, the right hand side of the above equation is constrained by a maximum and minimum risk value. Although these limits vary, the majority of consumable items at

SPCC are constrained between a minimum risk of 0.10 and a maximum risk of 0.35. The so-called basic reorder level R* is the solution to the constrained risk equation.

The optimality condition for Q is determined by taking the partial derivative of TVC(Q,R) with respect to Q and setting the result equal to zero. However, the resulting expression is difficult to solve explicitly for Q. In practice SPCC first determines Q using the deterministic economic order quantity equation:

$$Q^* = \sqrt{\frac{8AD}{IC}} \quad ,$$

where Q* is the called the unconstrained reorder quantity. SPCC then applies a series of constraints to Q* resulting in an initial constrained or basic reorder quantity Q_1. These constraints ensure that the order quantity is at least 1, that a sufficient quantity is ordered to ensure that the total procurement workload does not exceed the workload capacity of the purchasing department, and that the order quantity be no greater than six quarters' worth of demand. The latter constraint is a DOD mandated restriction.

Finally, additional constraints are applied to the basic reorder point R* and the basic reorder quantity Q_1 to obtain the final constrained reorder point $\hat{R}$ and reorder quantity $\hat{Q}$. The first of these final constraints ensures that a minimum reorder level is met, normally set to 0 or 1. Other constraints placed on the reorder level and reorder

quantity ensure that the safety levels are such that on-hand assets do not exceed the shelf life quantity[3] of the item.

D. FORECASTING QUARTERLY DEMAND IN UICP

Although NAVSUP Publication 553 [Ref. 21] provides a detailed description of forecasting in UICP, recent changes have rendered this document inaccurate in this regard. The recent changes in the demand forecasting process include the selection of various parameters used and the trend detection technique employed.

The UICP system generally uses single exponential smoothing to forecast mean quarterly demand and the mean absolute deviation of demand (MAD). At SPCC the smoothing constant for both forecasts is currently set at 0.1. One exception to this rule involves very low demand items where a power rule is used to forecast MAD. Other exceptions occur when a significant change in mean demand is identified based on the last quarterly observation or when recent observations indicate that demand is trending up or down.

Prior to actually computing the next quarterly demand forecast, the most recent quarterly demand observation is examined to determine if it falls within certain limits. This process, called "step" filtering, is used to determine if there has been a significant change in the mean, one that warrants discarding a majority of the historical demand data and computing the forecast using only recent data. If the

---

[3]The shelf life of an item is the life span of an item from the date of manufacture or inspection until the next inspection date for continued usefulness or disposal. The shelf life quantity is the expected quantity of demand to occur during the shelf life of an item. Shelf life items include batteries, chemicals, gaskets, etc.

process is "out-of-filter;" i.e., the last two observations have exceeded upper or lower control limits on the same side, the forecast is computed using only recent data. At SPCC a four-quarter average is used to forecast demand and a power rule based on this demand forecast [Ref. 23] is used to forecast MAD. If the observation is the first observation to exceed the limits, then the observation is ignored and the previous demand and MAD values are used. If the process is "in-filter" then the most recent demand observation is subjected to a trend detection test.

SPCC has implemented a Kendall trend detection test developed by Boyarski and Bissinger [Ref. 24]. This process uses a statistical test involving a "window" that contains recent observed data to determine the likelihood that demand is trending. The size of the observation "window" varies based on the mean and variability of demand. The statistical test employed varies based on window size and the variability of demand. If trending is detected then the next quarter's forecast is computed using only the recent data. In this case SPCC forecasts demand using a four-quarter average, while MAD is forecasted using a power rule. The following quarter the forecasting process returns to single exponential smoothing unless another step or trend is detected.

III. THE MODIFIED SILVER MODEL

A. INTRODUCTION

The modified Silver model is an extension of an inventory control model described by Edward Silver [Ref. 7] for probabilistic demand with a mean that varies significantly over time. A limitation of the Silver model which makes direct implementation in DOD systems impractical is the assumption that procurement lead time is fixed. The modified Silver model described extends Silver's model to accommodate variability in procurement lead time.[4] This variability is included in the determination of the reorder point.

Because DOD constrains the maximum order quantity to be no more than six quarters' worth of demand, the maximum order interval length is constrained in the modified Silver model to six quarters. The Silver-Meal heuristic used in Silver's model was changed to incorporate this constraint in the modified Silver model.

The remainder of this chapter describes the modified Silver model in detail. Section B provides underlying assumptions and defines notation. Section C is devoted to a discussion of the model. Section D provides some additional remarks concerning the estimation of parameters and model implementation.

---

[4]The issue of fixed lead times is as much a contracting issue as a model consideration. In many instances, procurement lead times can be considered nearly fixed. This is especially true when there are few vendors, special purchase agreements exist, or firm lead times are specified in the contract. Inclusion of lead time variation in the modified Silver model parallels, in many ways, its inclusion in the UICP model.

B. ASSUMPTIONS AND DEFINITIONS

The modified Silver model most closely resembles a periodic review (R,s,S) model where inventory position is checked every R time units.[5] If the inventory position (on hand + on order - backorders) is above the reorder point s, then no order is placed. If the inventory position is at or below s, then an order is placed to bring inventory position to level S [Ref. 9: p. 258]. However, in the classical model mean demand is assumed to be stationary, enabling the decision variables s and S to be computed and set for reasonably long periods of time (i.e., as long as no shift in mean demand is detected). For the modified Silver model mean demand is assumed to vary significantly over time, thus appropriate values of s and S would also be expected to change, perhaps with each review cycle. In a periodic review system, the selection of the value of the decision variable R, the time between reviews, generally corresponds to some logical time interval; e.g., week or month. In our case the decision variable R will be fixed at once a week, implying a policy of having a fixed time period between supply demand reviews.[6]

In addition to the key assumption that mean demand varies over time, the modified Silver model includes the following underlying assumptions:

---

[5]The reader is cautioned not to confuse the use of R in this section (time between reviews) and its use in Chapter II (reorder point for the UICP model).

[6]Until recently, SPCC conducted supply demand reviews on a bi-weekly basis, although reviews were sometimes run less frequently to postpone the expenditure of funds. As of 01 October 1994, SPCC conducts supply demand reviews on a monthly basis. This policy change was made to increase the time allowed for administrative reorder review. The selection of weekly reviews for the simulation implementation is to allow for the least amount of deviation of the UICP model from its continuous review assumption while still maintaining a periodic review system.

- As described above, calendar time is divided into fixed time periods of the same length. Reviews will be conducted at the end of each period and orders will arrive at the start of a period.

- The mean and standard deviation of procurement lead time are known or can be reasonably estimated.

- At any review instance, demand forecasts exist for the next N time periods, where N denotes the length of the forecast horizon.

- The selection of the value of s does not depend on the value of S to be used.

- Demand forecast errors over intervals of length L+1, where L is mean lead time, are Normally distributed with no bias (the significance of an interval of length L+1 will become apparent later in the discussion).

- An estimate of the standard deviation of the demand forecast error can be made for periods of length L+1.

- Holding costs are charged only on inventory carried from one period to the next.

- Safety stock will be determined based on a desired service level specified by a probability of no stockout during a replenishment cycle. Safety stock is the average level of net stock on hand just before a replenishment arrives. Safety stock provides a buffer or cushion against larger-than-expected demand during the replenishment lead time.

- Receipts of outstanding orders do not cross in time.

The following notation is used in the development of the model:

A - administrative order cost (dollars per order).
d_i - forecasted demand for period i (units per period).
h - holding costs (dollars per unit per period).
IP - inventory position at the time of review (t_0).
k_a - actual safety factor based on the current inventory position if an order is not placed.
k_r - required safety factor at the current review to attain the desired level of service for L+1 time periods.
L - mean procurement lead time (in periods).
Q - the size of the replenishment quantity.

P - desired probability of no stockout per replenishment cycle.
s_i - reorder point at time of review i.
T - integer number of periods that the current order is expected to cover.
t_0 - the time of the current review (time 0).
τ - random variable representing procurement lead time.
X1 - forecasted demand over the time interval t_0 to L+1.
X2 - forecasted demand over the time interval t_0 to T-1.
X3 - forecasted demand over the time interval T-1 to L+T.
σ_τ - standard deviation of procurement lead time
σ_{X1} - standard deviation of forecast error over the interval t_0 to L+1.
σ_{X2} - standard deviation of forecast error over the interval t_0 to T-1.
σ_{X3} - standard deviation of forecast error over the interval T-1 to time L+T.

Figure 3.1 provides a graphical representation of the various time intervals involved in the modified Silver model. Note, however, that X1, X2, and X3 refer not to the length of the associated interval but rather to the amount of forecasted demand over the associated interval.

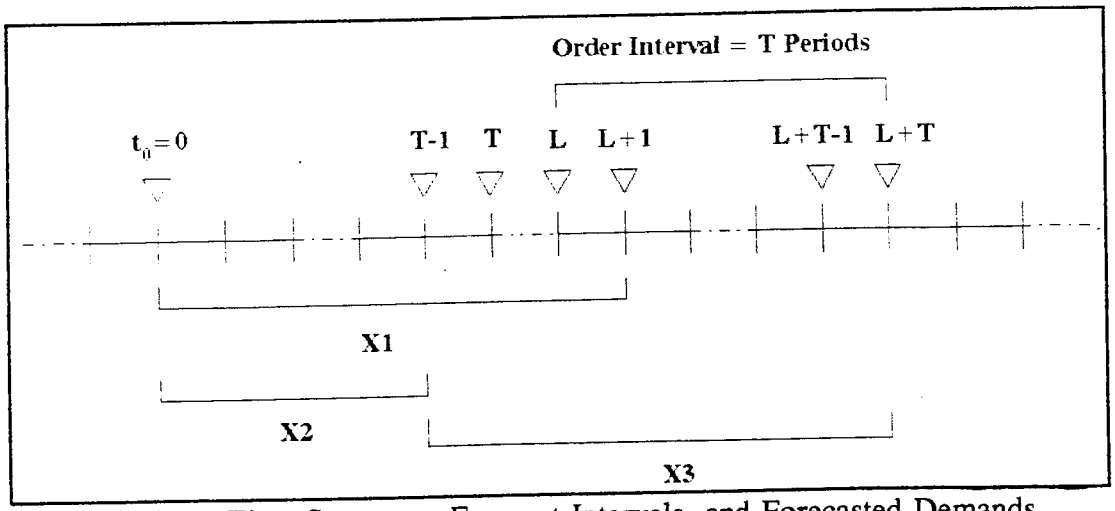


FIGURE 3.1. Time Sequence, Forecast Intervals, and Forecasted Demands.

C. MODEL DEVELOPMENT

1. Determination of the Reorder Point

In deciding whether to place an order at the current review (t_0) it must be determined if the inventory position (IP) is such that the specified service level will be met for the next L+1 periods. That is, if an order is not placed at time t_0, the current inventory position must provide adequate protection for a time interval of length L+1, which would be the expected time of receipt of an order placed at the time of the next review (t_0+1). Therefore, in determining the reorder point we are concerned about the expected demand X1.

Since forecast errors are assumed to be Normally distributed, the actual safety factor [Ref. 25: p. 365] for an interval of length L+1 is

$$k_a = \frac{IP-X1}{\sigma_{X1}} , \tag{3.1}$$

where IP is the inventory position at time t_0. The required safety factor (i.e, the safety factor necessary to provide the desired service level) depends only on the probability of no stock-out, P, specified by the inventory manager. Again, under the assumption that forecast errors are Normally distributed, the required safety factor k_r must satisfy

$$P(Z \geq k_r) = 1-P , \tag{3.2}$$

the probability that a standard Normal variable (Z) takes on value of k_r or larger

[Ref. 9: Chap 7]. Therefore, an order should be placed at the current review period if $k_a < k_r$; that is, if the actual safety factor is not sufficient to provide the desired level of service for the next L+1 periods.

Since the inventory position is known at t_0, the values which must be estimated in equation 3.1 are X1 and σ_{X1}. Forecasted demand, X1, is simply the sum of the forecasts for each individual period in the interval from t_0 to L+1. An estimate for the standard deviation of forecast errors over the interval is given by the following expression:

$$\sigma_{X1} = \sqrt{\sum_{i=1}^{L+1} \sigma_i^2 + \bar{d}_{X1}^2 \sigma_\tau^2} \qquad (3.3)$$

where i=1 is the first period following t_0, σ_i^2 is an estimate of the variance of the demand forecast error for the i^{th} period, $\bar{d}_{X1}$ is the average period demand over the interval associated with X1, and σ_τ^2 is the variance of procurement lead time. The derivation of equation (3.3) is provided in section D of this chapter.

2. Determination of the Order Interval (T)

The length of the order interval T is determined using the Silver-Meal heuristic, which selects T such that the total relevant costs per unit time for the duration of the replenishment quantity are minimized [Ref. 9: Chap 6]. The method is a heuristic in that it selects T corresponding to the first minimum which occurs. This minimum is not necessarily the global minimum. The heuristic selects the lowest integer value of T that produces a local minimum of the function

$$TRCUT(T) = \frac{A + h\sum_{i=1}^{T}(i-1)d_i}{T} \qquad (3.4)$$

where TRCUT is defined to be total relevant cost per unit time. It should be noted that in the implementation of the modified Silver model we select the value of T which minimizes TRCUT(T) from among the values 1 to the forecast horizon (6 quarters). This is accomplished by computing TRCUT(T) for each of the values 1 to the forecast horizon, and selecting that value of T corresponding to the smallest TRCUT(T). The modification is an improvement over the heuristic in that it guarantees the minimum over the forecast horizon, whereas the heuristic does not. This modification in the heuristic was implemented because of the DOD constraint which limits the maximum reorder amount to the expected demand over 6 quarters.

3. Determination of the Order Quantity

Determination of the order-up-to-level (S), and hence the order quantity (Q), is a function of the length of the order cycle (T). Two distinct cases exist; one if T=1 and the other if T>1. The differences between these cases will become apparent from the following discussion.

The case for T=1 is represented graphically in Figure 3.2. If T=1, then we are planning the current replenishment to have only enough stock to meet our specified service level through the first period following receipt of the order (i.e., period L+1). This is precisely the value of X1 which was calculated in the determination of the reorder point. In this case we simply order the deficit to the

current inventory position (IP) necessary to achieve the specified service level. Equation (3.5) provides the value of the order quantity in this situation.

$$Q = X1 + k_r \sigma_{X1} - IP \ . \tag{3.5}$$

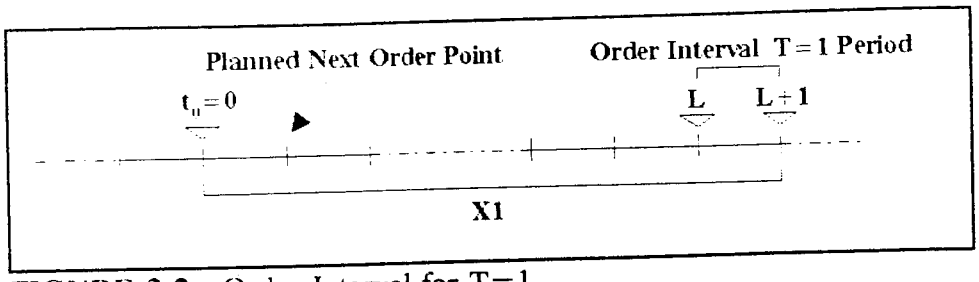


FIGURE 3.2. Order Interval for T=1.

The case for T > 1 is represented graphically in Figure 3.3. If T > 1, then we are planning on the current order providing sufficient coverage for T periods after receipt of the replenishment; i.e., we are planning not to place an additional order until time T assuming that τ has a stationary mean, L. To reduce the possibility that a reorder will occur at an earlier time period, say T-1, we can introduce a small

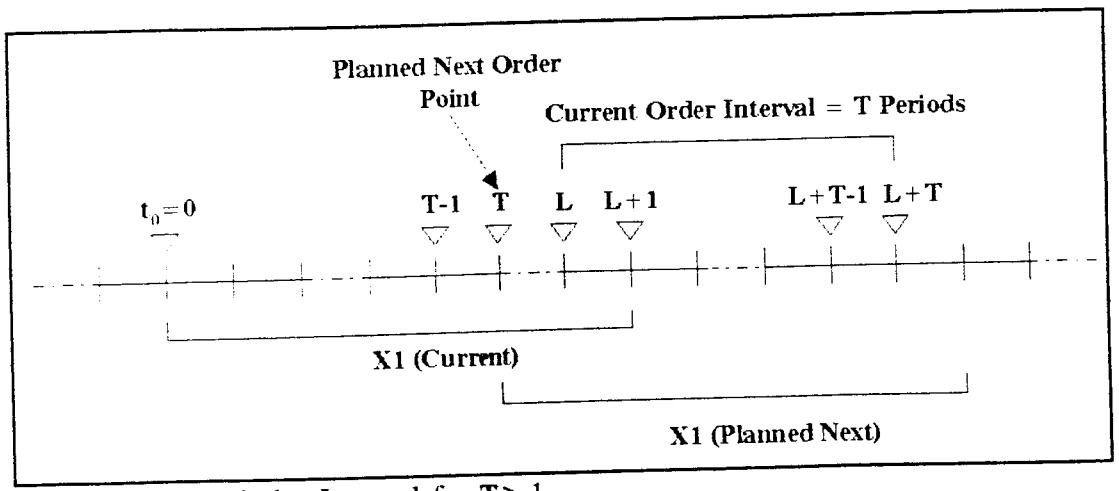


FIGURE 3.3. Order Interval for T > 1.

buffer [Ref. 7: p. 375] to the estimated inventory position at time T-1. As is usually the case, buffer or safety stock is expressed as some multiple (denoted by b) of the measure of uncertainty of forecast errors over the interval of concern.[7] In this case we are concerned with the interval from the present, t_0, until T-1, which has the expected demand X2 (see Figure 3.1). Therefore it is reasonable to express this buffer as a multiple of σ_{X2}. In addition, we are assuming that mean demand varies over time, we should also expect the reorder point at time T-1 to be different from the current reorder point. The above argument can be expressed symbolically as:

$$s_{T-1}+b\sigma_{X2} = IP+Q-X2 \tag{3.6}$$

where:

IP is the present inventory position (time t_0);
Q is the size of the order quantity at time t_0;
X2 is forecasted demand over the interval t_0 to T-1;
s_{T-1} is the reorder point at time T-1;
σ_{X2} is the standard deviation of forecast error over the interval X2;
b is the coefficient of additional buffer stock for the interval X2.

Solving equation (3.6) directly for Q yields the following expression for the reorder quantity:

$$Q = s_{T-1}+b\sigma_{X2}+X2-IP\ . \tag{3.7}$$

---

[7]Inclusion of a buffer quantity obviously has tradeoffs; i.e., additional carrying costs versus the risk of additional ordering and shortage costs. In practice, the amount of buffer should be a management decision or "weighing" of these tradeoffs. Silver recommends a small buffer value, if any [Ref. 7:p. 375]. Our investigation via simulation indicates a very small penalty in terms of total costs for small buffer values (b=0 to 0.9). Additional comments concerning selection of this buffer are provided in Chapter IV.

Since the required safety factor k_r at any time is dependent only on the specified service level, from equations (3.1) and (3.2) it follows that the reorder point s_{T-1} can be expressed as:

$$s_{T-1} = X3 + k_r \sigma_{X3} \tag{3.8}$$

where X3 is the forecasted demand over the interval from time T-1 to L+T. Substituting equation (3.8) into equation (3.7) yields the order quantity equation for the case when T > 1:

$$Q = X3 + k_r \sigma_{X3} + b\sigma_{X2} + X2 - IP \ . \tag{3.9}$$

An intuitive explanation of this equation follows. The quantity X2+X3 represents the forecasted mean demand during the interval t_0 to L+T. To this quantity we add safety stock, some due to the variability of demand during the interval t_0 to T-1 and some due to the variability of demand during the interval T-1 to L+T. Since we currently have IP units of stock, the order quantity (Q) is the difference between this sum and our current value of IP.

D. PARAMETERS AND IMPLEMENTATION

1. Standard Deviation of Forecast Errors

Implementation of the modified Silver model requires estimation of three standard deviations of forecast error corresponding to the demand forecast intervals X1, X2, and X3, namely σ_{X1}, σ_{X2}, and σ_{X3}. Several techniques exist for estimating the

standard deviation of forecast error for demand intervals when the intervals are fixed and mean demand is constant [Ref. 25: p. 366-368]. In the special case where the interval is a procurement lead time which is probabilistic, and the distribution of demand about its mean is independent of the distribution of lead time about its mean, the standard deviation of lead time demand can be estimated by:

$$\sigma_{LTD} = \sqrt{L\sigma_1^2 + D^2\sigma_\tau^2} \tag{3.10}$$

where: σ_{LTD}=standard deviation of lead time demand;
L=mean lead time;
D=estimate of mean demand (forecast) for one period;
σ_1^2=estimated variance of forecast errors about the forecast D;
σ_τ^2=estimated variance of lead time.

This is the equation currently used by SPCC to estimate lead time demand for items with moderate to high mean demand. In the modified Silver case where forecast intervals have different lengths and mean demand varies over time, equation 3.10 cannot be used directly, although a similar estimate can be developed.

To obtain an estimate for the standard deviations of forecast errors, we first assume that the coefficient of variation, defined as the ratio of the standard deviation of forecast error of a single period to its mean (forecast), is constant over the forecast horizon [Ref. 7: p. 15]. This relationship can be expressed as follows:

$$c = \frac{\sigma_1}{d_1} = \frac{\sigma_i}{d_i} \tag{3.11}$$

where: c=coefficient of variation;
σ_1=estimate of the standard deviation of forecast error for the current period;
d_1=current mean demand (forecast);
σ_i=estimate of the standard deviation of forecast error for the i^{th} period within the current forecast horizon; and
d_i=estimated demand (forecast) for the i^{th} period within the current forecast horizon.

Alternatively, equation (3.11) can be expressed as:

$$\sigma_i = cd_i \tag{3.12}$$

for the i^{th} period of the forecast horizon. In the case where procurement lead time is fixed and forecast errors in consecutive periods are assumed to be independent, equation (3.12) can be used to develop the following estimates for the standard deviations of forecast errors for the periods associated with X1, X2, and X3 [Ref. 7: Appendix B]:

$$\sigma_{X1} = c\sqrt{d_1^2+d_2^2+\ldots+d_{L+1}^2} \tag{3.13}$$

$$\sigma_{X2} = c\sqrt{d_1^2+d_2^2+\ldots+d_{T-1}^2} \tag{3.14}$$

$$\sigma_{X3} = c\sqrt{d_{T-1}^2+d_T^2+\ldots+d_{L+T}^2}\ . \tag{3.15}$$

Since the modified Silver model assumes that lead times are stochastic and the demand forecasts X1 and X3 occur over an interval that contain a mean lead time period (both are of length L+1), a slightly more complicated approach must be

employed to estimate σ_{X1} and σ_{X3}. This will be accomplished by first estimating the variance (and thus the standard deviation) of demand over a lead time and extending this estimation to an interval of length L+1.

Using the conditional formula for variance [Ref. 26: Chapter 3], the variance of lead time demand can be expressed as:

$$Var(\sum_{i=1}^{\tau} d_i) = E[Var(\sum_{i=1}^{\tau} d_i|\tau)] + Var[E(\sum_{i=1}^{\tau} d_i|\tau)] \qquad (3.17)$$

where τ is the random variable representing lead time, with known mean L and variance σ_τ^2. A reasonable estimate for the expected value in the second term in the right hand side of equation (3.17) is $\bar{d}_L\tau$, where $\bar{d}_L$ is the average of the d_i, i =1 to L. Substituting this estimate into equation (3.17) and simplifying the first term in the right hand side under the assumption that forecast errors in consecutive periods are independent, yields:

$$Var(\sum_{i=1}^{\tau} d_i) = E(\sum_{i=1}^{\tau} \sigma_{d_i}^2) + Var(\bar{d}_L\tau) \; . \qquad (3.18)$$

Since $\bar{d}_L$ is a constant, the second term in the right hand side of equation (3.18) can be further simplified as:

$$Var(\sum_{i=1}^{\tau} d_i) = E(\sum_{i=1}^{\tau} \sigma_{d_i}^2) + \bar{d}_L^{\,2}\sigma_\tau^2 \; . \qquad (3.19)$$

The first term on the right hand side of equation (3.19) is somewhat more difficult to approximate. One way to approximate its value is through a Taylor expansion

[Ref. 27: p. 30-31]. Since τ is a random variable with known mean L and variance σ_τ^2, the second order Taylor expansion about the mean of the function

$$f(\tau) = \sum_{i=1}^{\tau} \sigma_{d_i} \tag{3.20}$$

is

$$f(\tau) = f(L) + f'(L)(\tau - L) + e \tag{3.21}$$

where e represents the error term. Ignoring the error term and taking the expected value of equation (3.21) yields:

$$E[f(\tau)] \approx f(L) . \tag{3.22}$$

Substituting this result into equation (3.19) yields the following result as a representation of lead time demand:

$$Var(\sum_{i=1}^{\tau} d_i) = \sum_{i=1}^{L} \sigma_{d_i}^2 + \bar{d}_L^2 \sigma_\tau^2 . \tag{3.23}$$

We are actually interested in the variance of demand for an interval of length $\tau+1$. Recognizing that this is simply the sum of a random variable and a constant, the same argument presented above can be used to yield the following approximation for demand over an interval of length $\tau+1$:

$$Var(\sum_{i=1}^{\tau+1} d_i) = \sum_{i=1}^{L+1} \sigma_{d_i}^2 + \bar{d}_{L+1}^2 \sigma_\tau^2 \ . \qquad (3.24)$$

where $\bar{d}_{L+1}$ is the average of the d_i, i=1 to L+1. Rewriting equation (3.24) in terms of the intervals associated with X1 and X3, and expressing the results as a standard deviation, yields the following:

$$\sigma_{X1} = \sqrt{\sum_{i=1}^{L+1} \sigma_{d_i}^2 + \bar{d}_{X1}^2 \sigma_\tau^2} \qquad (3.25)$$

and

$$\sigma_{X3} = \sqrt{\sum_{i=T-1}^{L+T} \sigma_{d_i}^2 + \bar{d}_{X3}^2 \sigma_\tau^2} \qquad (3.26)$$

where $\bar{d}_{X1}$ and $\bar{d}_{X3}$ are average demand over the respective forecast intervals. Substituting the result of equation (3.12) into the first term of the right hand side of equations (3.25) and (3.26), gives the following:

$$\sigma_{X1} = \sqrt{c^2 \sum_{i=1}^{L+1} d_i^2 + \bar{d}_{X1}^2 \sigma_\tau^2} \qquad (3.27)$$

and

$$\sigma_{X3} = \sqrt{c^2 \sum_{i=T-1}^{L+T} d_i^2 + \bar{d}_{X3}^2 \sigma_\tau^2} \ . \qquad (3.28)$$

Equations (3.14), (3.27) and (3.28) are the equations which will be used to estimate the standard deviations of forecast error in the modified Silver model.

2. Estimating Demand Variability

Being able to reasonably estimate demand variability in stochastic inventory models is essential to the setting of safety levels. Over-estimation results in larger than necessary safety stocks and associated inventory costs, while under-estimation results in lower service levels than desired. In the modified Silver model demand variability is expressed in terms of the coefficient of variation (denoted as c). Although Silver provides guidance in the determination of this value [Ref. 7: p. 376-377], none of the methods specified would be practical for implementation in existing DOD systems given the large number of line items being managed. Rather, it would be desirable to have a dynamic method to estimate variability which would make use of existing data systems and time series analysis. To this end the modified Silver model, as implemented in the computer simulation, defaults to the use of forecast mean absolute deviation (MAD) to estimate the coefficient of variation (c).

The relationship between forecast MAD and demand variability when forecast errors are assumed to be Normally distributed has long been accepted within the Navy's inventory control system. Under the Normality assumption, the constant of proportionality of MAD to the standard deviation is approximately 0.8 [Ref. 25: p. 282-283]. Alternatively, the relationship can be stated as:

$$\sigma_1 = 1.25MAD \ . \tag{3.29}$$

The practice of using equation (3.29) to estimate standard deviation has been shown to be valid for most of the Navy's demand classes [Ref. 28: p. 1]. The primary exception to this approximation is for low demand items.

Since a primary assumption of the modified Silver model is that forecast errors are Normally distributed, a reasonable estimate for the coefficient of variation c can be obtained by combining equations (3.11) and (3.29) to obtain:

$$c = \frac{1.25MAD_1}{d_1} \tag{3.30}$$

where MAD_1 is the forecast MAD for the next period and d_1 is the next period forecast. Equation (3.30) is used in the modified Silver model to estimate the value of c, which is then used in computing the standard deviations of forecast error for future periods out to the forecast horizon.

3. Normality Assumption for Forecast Errors

The assumption that forecast errors are Normally distributed or at least approximately Normal is reasonable in most cases [Ref. 25: Chap 19]. Since this assumption is fundamental to the modified Silver model, it would be appropriate to present some empirical evidence in support of it. With the exception of very low demand items, the Normality assumption is also a key assumption in the existing UICP model in the estimation of the variance of forecast errors.

In order to test the Normality assumption for a subset of the stationary mean demand streams later used in analyzing the performance of the model, a computer simulation was written which generated quarterly demand observations and forecasts, and computed forecast errors. The forecasting procedure is a replication of SPCC's forecasting procedure. Data was collected for 10 replications of 100 quarters each (1000 data points) for each demand classification. These errors were then analyzed for Normality using IBM Corporation's "A Graphical Statistical System" (AGSS). Table 3.1 provides a summary for low to high demand cases with varying degrees of variability generated from a Normal distribution.[8] Table 3.2 provides a summary of the analysis for several very low and low demand cases generated from a Poisson distribution. In each table column one provides the mean assumed to generate the demand stream. Column two of Table 3.1 gives the assumed variance of demand. The remaining columns in each table provide distribution data for forecast errors. Test results (p values) for the Chi-Square, Kolmogorov-Smirnov and Cramer-von Mises goodness of fit tests are provided. An attained p value of less than 0.01 for any test indicates the lack of a good fit. The data strongly supports the Normality assumption when demand is generated from a Normal distribution, with the exception of very highly variable demand. In the cases of very low and low demand Poisson data the Normality assumption is even less accurate. The empirical results confirm

---

[8] Since the distribution of forecast errors is a convolution of the demand distribution and the forecast distribution, Normally distributed demand does not obviously lead to Normally distributed forecast errors. In the case of linear, discrete, time-invariant systems such as that being tested here, it can be shown that the forecast errors will also be Normally distributed [Ref. 25: p.275-278].

our intuitive expectations and the observed results reported elsewhere in this thesis. In the case of demand generated from a Normal distribution, the underlying distribution is actually a truncated Normal distribution since demand cannot be negative. When the mean is close to zero or the variance is high, truncation is more prevalent. Since the observed data distribution is skewed to the right of its mean, we would expect forecast errors to display a similar skewness. In the case of demand generated from a Poisson distribution, which is characteristically skewed to the right of its mean, one would expect errors to be similarly skewed. The empirical results reported in the Tables 3.1 and 3.2 support these expectations.

TABLE 3.1. NORMAL (TRUNCATED) DEMAND AND FORECAST ERROR

Mean Demand	Var Demand	Mean Error	Sigma Error	Skew	Kurt	Chi-Sq	K-S	C-V
2.0	2.0	0.061	1.484	0.231	2.806	0.028	0.267	>.15
4.0	2.6	0.067	1.742	0.106	2.978	0.455	0.732	>.15
4.0	10.2	-0.044	3.324	0.327	2.737	6.0xE-5	0.128	<.15
4.0	31.4	0.053	5.010	0.567	2.891	0.0	2.4xE-6	<.01
12.0	23.0	-0.030	5.193	-0.044	2.991	0.375	0.464	>.15
12.0	92.0	0.053	9.252	0.280	2.562	4.8xE-7	0.099	<.15
12.0	282.0	1.160	14.970	0.581	3.311	0.0	0.0	<.01
25.0	100.0	-0.198	10.919	0.020	2.994	0.960	0.950	>.15
25.0	400.0	-0.096	19.760	0.340	3.160	0.010	0.251	>.15
25.0	1225.0	0.825	30.96	0.585	3.187	0.0	0.0	<.01

TABLE 3.2. POISSON DEMAND AND FORECAST ERROR

Mean Demand	Mean Error	Sigma Error	Skew	Kurt	Chi-Sq	K-S	C-V
0.25	-0.011	0.485	1.747	6.372	0.0	0.0	<.01
0.75	-0.014	0.951	1.046	4.759	0.0	0.0	<.01
1.0	0.010	1.070	0.804	3.658	0.0	2.1xE-6	<.01
1.5	0.039	1.328	0.658	3.942	0.0	5.9xE-5	<.01
2.0	0.073	1.557	0.526	3.548	4.2xE-7	0.002	<.01

4. Implementing Periodic Reviews

Until this point in the discussion of the modified Silver model, the review periods have been implicitly assumed to coincide with forecast intervals. That is, in order to explain the mechanics of the model, we have assumed that the periodic reviews have occurred at the start of a forecast period. This is not necessarily true in reality. In the case of the Navy's inventory control system, reviews generally occur every one, two or four weeks. Forecasting, on the other hand, is done quarterly. This section discusses the adjustments that have to be made to the model to accommodate this situation. We will examine two cases: one when $T > 1$ and the other when $T = 1$. In each case we assume that reviews are conducted on a weekly basis, although the same procedures can be used if reviews are conducted less frequently. For the purpose of this study each quarter is assumed to have 13 weeks.

When $T > 1$, forecast estimates are made as previously described with the exception that partial forecast period data is used. Figure 3.4 provides a graphical representation of this case. For example, if a review is held the 8th week of a quarter when quarterly forecasting is employed, the demand forecast for the first period, d_1,

will be 5/13^{ths} of the current quarter's forecast and 8/13^{ths} of the next quarter's forecast. The second period forecast, d_2, will be calculated in a similar manner as 5/13^{ths} of the next quarter's forecast and 8/13^{ths} of the following quarter's forecast. This procedure is continued until the appropriate number of intervals are collected; e.g., L+1 periods for X1. The same technique is used in determining the order interval. Note that if an order is placed at time t_0, then the interval associated with the demand forecast X1 for each subsequent weekly review falls within the order interval of the order placed at time t_0 until time T-1 is reached. This important recognition leads to the significant difference in the model's implementation when T=1.

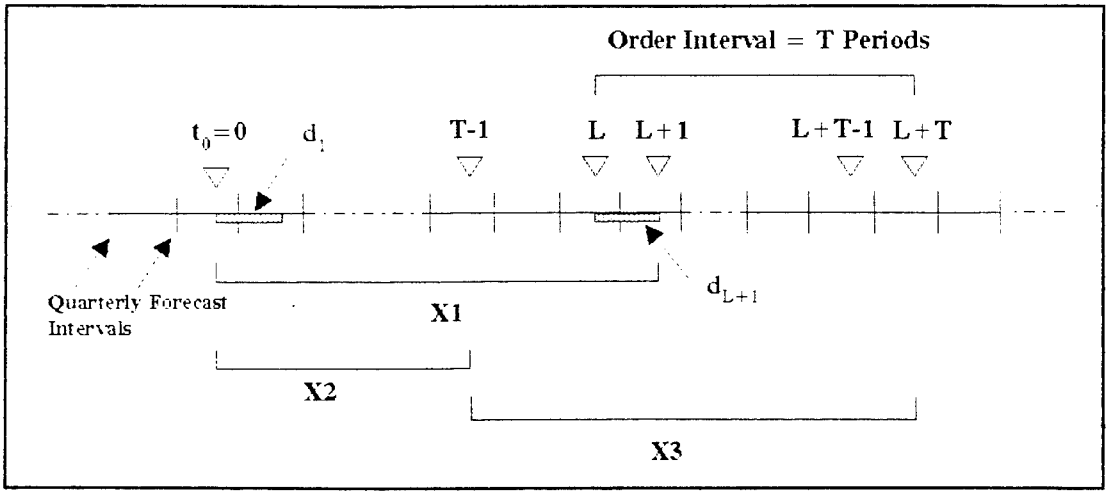


FIGURE 3.4. Review Periods Between Forecast Intervals (T > 1).

Figure 3.5 provides a graphical representation of the special case when T=1. The first time an order is placed for an interval of length T=1, at time t_0, the procedure is applied in a similar fashion as indicated when T > 1. However, recall

when T=1 the only demand forecast computed is X1. Then, again assuming reviews are conducted on a weekly basis, for the next 12 reviews the review interval is fixed to cover demand over the same time interval t_0 to L+1 from the previous order. That is, the value of X1, from t_0 to L+1, is simply updated at each review to reflect actual demand which has occurred since time t_0. If the actual demand has been larger than expected, an incremental order is generated to increase the order quantity of the previous order. It is assumed that such incremental orders will occur infrequently since buffer stock due to the variability of demand has already been included in the computation of X1. Also, it is assumed that such incremental orders would be small and that a change in the order quantity would be accepted by the procurement office if in a pre-award stage, or by a contractor if a contract has already been issued. This provision is necessary to prevent excessive ordering. In the case where T=1, a subsequent review over a new interval of length L+1 would look beyond the order

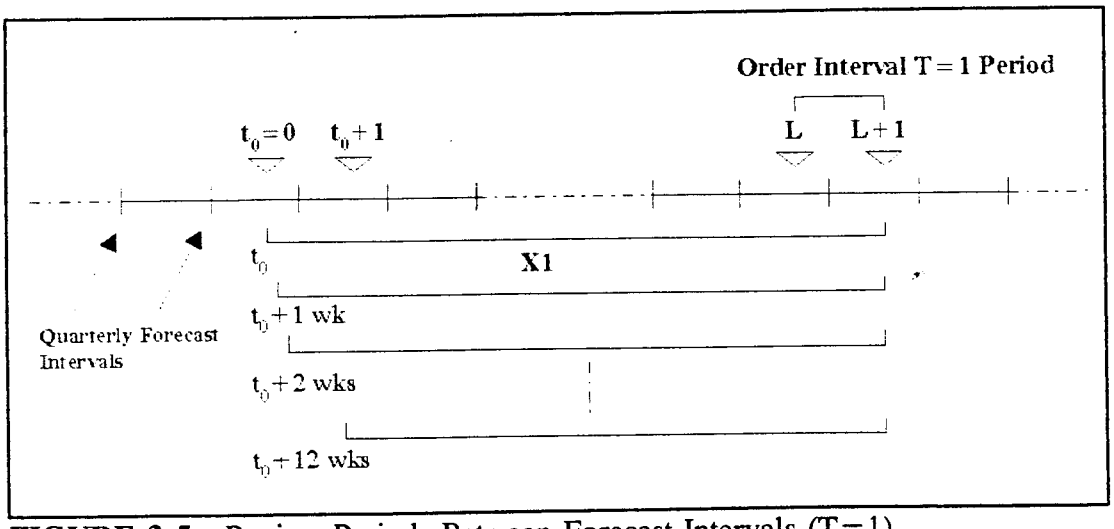


FIGURE 3.5. Review Periods Between Forecast Intervals (T=1).

interval of the last order, making it highly likely that a small order would be placed at each review. Ideally, we do not wish to order again until time t_0+1 quarter. Recall that in the basic model when we wished to reduce the probability of an order occurring prior to the expected time of the next order, we included a small buffer value in the order quantity computation. In this special case an analogous quantity is also included in the order quantity computation. Here the same multiplier (b) is used with an estimate of variability representing $1/13^{th}$ of the quarterly variance.

Some final notes concerning this special case. First, an order cycle of $T=1$ only occurs when the dollar value of forecasted demand is high, thus the number of items falling into this category should be limited. For example, under a steady state assumption, if forecasted demand for an item is 4 units per quarter, the administrative ordering cost is \$850, and the annual holding cost rate is 0.23, then the unit cost of the item must be greater than \$3,696 to have an order interval of length $T=1$ (see Section 5 below). Secondly, an alternative approach was considered in which the 12 weekly reviews following an order placed for an interval of length $T=1$ were suppressed to prevent additional orders. Simulation test results using this procedure indicated very little difference in the total number of orders from the procedure implemented above.[9]

[9] In the simulation, incremental orders using the implemented procedure were counted as orders in the count of total orders.

5. Estimating the Order Interval Length (T)

It is often useful to be able to estimate the length of the period that orders will cover for given values of the system parameters. Since the order interval length (T) for the Silver model is based on the Silver-Meal heuristic, the length of the order cycle is a simple function of holding and administrative ordering costs and forecasted demand. The behavior of the heuristic is easily analyzed in the case where mean demand does not vary. This analysis is useful when making comparisons to the steady state EOQ model.

Recall that the objective of the Silver-Meal heuristic is to select the lowest integer value of T that produces a local minimum of the function

$$TRCUT(T) = \frac{A + h\sum_{i=1}^{T}(i-1)d_i}{T} \tag{3.31}$$

where TRCUT is defined to be total relevant costs per unit time. Assume that the holding costs (h) and administrative ordering costs (A) are fixed, and mean forecasted demand (D) does not vary. An algebraic expression to determine the minimum value of T when TRCUT(T) exceeds TRCUT(T-1) is given by the following inequality:

$$\frac{A + h\sum_{i=1}^{T}(i-1)D}{T} \succ \frac{A + h\sum_{i=1}^{T-1}(i-1)D}{T-1} . \tag{3.32}$$

Rearranging this expression in terms of T and substituting IC/4 (annual holding costs expressed as a quarterly value) for the holding costs (h) results in:

$$T(T-1) > \frac{8A}{ICD} \tag{3.33}$$

which can be used to determine the length of the order cycle T for given system parameters C, A and I, and a given stationary mean demand rate D.

Equation (3.33) can also be used to show the relationship between the Silver-Meal heuristic and the EOQ when mean demand is stationary. Using the same notation used throughout this chapter, recall from Chapter II the basic EOQ is given by the following equation:

$$Q = \sqrt{\frac{8AD}{IC}} \, . \tag{3.34}$$

The equation for the length of the order cycle is T=Q/D. Substituting equation (3.34) for Q then gives the following equation for the EOQ order interval:

$$T = \sqrt{\frac{8A}{ICD}} \, . \tag{3.35}$$

By comparing equations (3.33) and (3.35) we can see the approximate equivalent relationship between the discrete Silver-Meal heuristic and the continuous EOQ model for the order interval.

IV. SIMULATION

A. INTRODUCTION

To analyze the performance of the modified Silver model, a discrete event Monte Carlo simulation was developed in two parts. In the first part, simulation code was written to represent the UICP inventory control system.[10] In the second part, the UICP simulator code was copied and modified, replacing the UICP "levels" setting program with the modified Silver model. The modifications included changes to the forecast system to provide multi-period forecasts. The software was written in Turbo PASCAL, Version 7.0 for IBM compatible personal computers.

Both simulations approximate the inventory management of a single consumable item for as many as 120 quarters. The user may choose to have quarterly demand data randomly generated using a Normal or Poisson distribution. The simulation allows the user to specify run characteristics, system parameters, and demand profiles. Several output options are available. An explanation of the basic functions of the simulation models are provided in the remainder of this chapter. Appendix A provides a complete listing of program code.

---

[10] The simulation code was co-developed by the author and Lieutenant Commander Donald C. Miller, a U. S. Navy Officer and graduate student at the Naval Postgraduate School studying operations research.

B. ASSUMPTIONS

The UICP model computes both system measures of effectiveness and shortage costs on a requisition basis. Although conceptually there is no restriction to having varying requisition sizes, computationally it requires that the distribution of requisition size be known. Additionally, maintaining time-weighted shortage statistics on a requisition short basis is considerably more complicated when partial issues of requisitions are allowed. To avoid these complications, it is assumed that each requisition is for a single unit. Under this assumption, time-weighted units short and time-weighted requisitions short are equivalent.

As discussed in Chapter III, although the Navy's UICP model is a continuous review model, inventory reviews are actually done on a periodic basis, generally once every two weeks or when adequate funds or computer time are available [Ref. 21: Chapter 3]. Since this type of uncertainty is difficult to model, it is assumed for modeling purposes that inventory reviews are held on a weekly basis. The same assumption is made in the modified Silver model.

Simulation time has historically been measured using one of two approaches. The first approach is called the *next-event time advance*, where future events are maintained on a calendar and simulation time is advanced to the next event. The second approach is called *fixed-increment time advance*, where time is advanced a pre-specified time increment, independent of events [Ref. 29: Chapter 1]. In modeling an inventory control system there are two primary events, issue and receipt of material. In reality, these events occur on a near continuous calendar. Since the actual arrival

distributions of these events are unknown (i.e., Navy data about the time between events is extremely difficult to obtain), the fixed-increment time advance approach is used. Under this approach the simulation clock is updated every Δt time units and a check is made to determine if any events have occurred during the previous interval. If events have occurred they are assumed to have occurred at the end of the interval. System states and statistics are updated accordingly.

Under this approach two considerations must be addressed. First, processing all events in an interval as if they occurred at one instance in time reduces the accuracy of the statistics gathered. Secondly, when two or more events occur during an interval, a set of rules must be developed which specifies the order in which the events are to be processed. Such rules may also lead to inaccuracies in the model's measurement of reality. These problems can be made less severe by selecting a small Δt time interval. Although a Δt of one day would provide the greatest accuracy, the required data structures are very large and execution time would be very long. For this reason, a relatively small Δt (one week) was chosen. When two or more events occur during the same interval, receipts are assumed to occur first, followed by issues, then ordering.

C. SIMULATION STRUCTURE AND DESIGN

The simulation programs are modular in design. Common functions and procedures are organized in self contained packages called *units.* Program unique processes are coded as program functions or procedures. The main program of each

simulation acts as a "control loop," making sequential calls to procedures and maintaining overall simulation flow.

The program units common to both the UICP simulator and the modified Silver simulator (MOD Silver) are *toolbox*, *unirand* and *pqueue*. *Toolbox* contains useful query and input/output (I/O) routines, as well as several statistical functions used to compute means, variances and confidence intervals. *Unirand* contains the random number generator and algorithms for generating probability distributions. These will be discussed in more detail in the next section. *Pqueue* is used exclusively to maintain outstanding stock orders and backordered requisitions. This event list is maintained as a priority queue using a heap data structure [Ref. 30: Chapter 7]. The UICP simulator makes use of one additional unit, *pdunit*, which contains procedures that enable the simulation program to interface with SPCC's PC-versions of PD-82 and PD-86. These latter two programs are written in COBOL and require extensive I/O record layouts.

Both simulations contain common functions and procedures, some of which are tailored to reflect unique parameter specifications. The common program functions and procedures include:

RunType:
A user interface procedure for entering simulation run specifications. These specifications include the number of replications, the number of quarters per replications, and the probability distribution to be used to generate demand.

Forecast:
A procedure which replicates SPCC's forecasting procedures.

GetMarkCode:

A function which determines the item mark code. This is a code based on forecasted quarterly demand and unit price. Mark code assignments are made by UICP and affect the forecasting and the levels setting technique used for an item [Ref. 21: Chapter 3].

LoadObserv:

A procedure used to generate demand and demand profiles.

SDR:

The supply demand review process, a procedure which determines whether or not a reorder should be placed.

Both LoadObserv and Forecast incorporate routines from SPCC's Demand Forecast Simulation [Ref. 31], a PC-based FORTRAN simulator developed by SPCC for demand and forecast analysis. The last three procedures, LoadObserv, Forecast and SDR will be discussed in detail in separate sub-sections below.

Unique to the UICP simulation is a procedure called *LoadLevels*. Its primary purpose is to determine the quarterly reorder quantity (Q) and reorder point (R) values using the UICP computational procedures. In the modified Silver simulation these quantities are determined in *SilverModel*, a sub-procedure of SDR.

There are also several other procedures and functions in each simulation which are used primarily for collection of statistics, report generation, and I/O processing. These include *InitializeStatArrays*, *InitializeArrays*, *PrintHeader*, *CalcSimStats*, *DisplaySimStats* and *DisplayQtrSimStats*.

1. Demand Observations

In order to simulate the actual quarterly demand patterns experienced by SPCC, random demand patterns were generated using a uniform random number

generator applied to either a Normal distribution for moderate to high demand items or a Poisson distribution for low demand items. The random number generator used was a prime modulus multiplicative linear congruential generator (modulus 2^{31}-1), based on a generator by Marse and Roberts [Ref. 29: p. 447]. The generator can produce up to 21,474 unique streams of 100,000 random numbers each. The simulation code will allow the user to specify up to 20,000 such streams. Each demand stream for a set of parameters is called a replication. Summary or simulation statistics are collected across all replications for a specified set of parameters.

The "polar method" [Ref. 29: p. 491] is used to transform the uniform random numbers into standard Normal random variates (denoted here as X). Normally distributed numbers (X') corresponding to the user specified mean (μ) and variance (σ^2) are computed by the transformation $X' = \mu + \sigma X$. Since demand is integral, a 0.5 rounding rule is employed. The algorithm used to generate Poisson (λ) random variates is based on the procedure of Law and Kelton [Ref. 29: p.503] which involves summing Exponential ($1/\lambda$) random variates.

Since events are processed on a weekly basis, quarterly demand is randomly distributed to occur weekly throughout the quarter. This is accomplished by associating with each unit of the quarterly demand observation a randomly generated uniform integer ranging from 1 to 13, corresponding to the week in the quarter in which that unit of demand occurs. The individual demanded units for each week are summed for each of the 13 weeks giving the weekly demand observations.

Since the principal concern of this study is the performance of the two models under non-stationary mean demand profiles, the user can specify up to 10 steps or trend periods per demand stream. In each step or trend period the mean used to generate demand is changed. If demand is being generated by a Normal distribution, the variance is correspondingly transformed to maintain the same ratio of the standard deviation to the mean (coefficient of variation), thus preserving the level of variability that was initially specified. A step in the mean is simply a point where the current quarterly mean is either increased or decreased by a non-negative multiplier. Symbolically, if D_t is the current mean, then $D_{t+1}=A * D_t$, where A is non-negative constant. The new mean remains in effect until another step or trend changes its value.

The trend function is exponential, allowing the user to specify a full range of convex, concave or linear patterns of growth or decline. Symbolically, the trend function is of the form $D_t=D_0*(1+A*t_i^{B})$ where D_t is the mean demand for period t, D_0 is the mean demand of the initial trend quarter, t_i is the *number* of quarters into the current trend period, and A and B are the specified trend parameters [Ref. 31]. Selecting a trend exponent parameter (B) of one results in a linear trend that has slope A. Figures 4.1 and 4.2 show graphs of trends produced for two different selections of the trend parameters A and B.

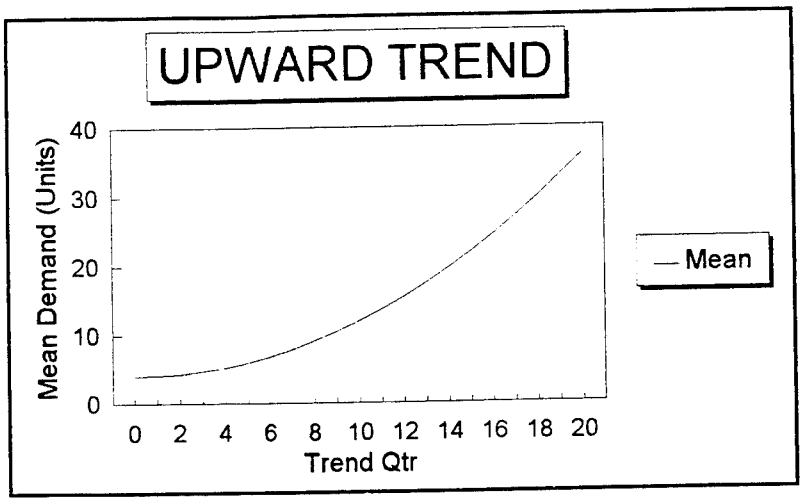


FIGURE 4.1. Increasing Demand; A=.02, B=2.

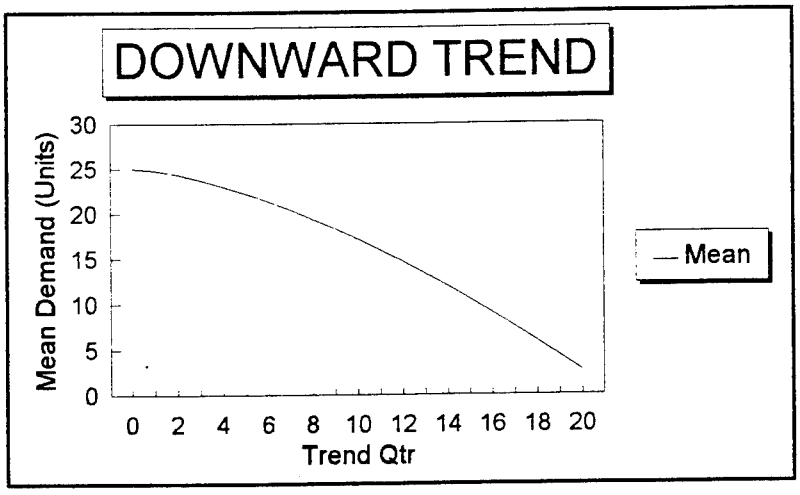


FIGURE 4.2. Declining Demand; A=-.01, B=1.5.

2. Forecasting

The simulation forecasting routine, *Forecast*, emulates the current forecasting methods used in UICP to forecast the next quarter's demand. This system was described in Chapter II.

Since the modified Silver model is capable of using forecast data beyond one quarter, the forecasting system had to be modified to generate this future forecast information. Since all steps and trends are specified prior to actually running the simulation, this information can be used to generate future forecast. To avoid providing unrealistic, "perfect" forecasts, all future forecast are based on the current single quarter forecast provided from the UICP forecasting system. The subsequent forecasts are the product of the current UICP forecast and the ratio of the mean used to generate the future quarter's demand and the mean used to generate the mean of the current forecast quarter. Therefore, if the forecast system has over or under estimated the current forecast quarter's demand, all future projections will similarly be high or low. Forecast are made each review cycle for the entire forecast horizon (mean lead time + 6 quarters). If no trends or steps are specified, the current forecast is used for each quarter within this horizon. This technique of generating future forecasts is analogous to an item manager making future projections based on advance knowledge of program changes which will result in an increase or decrease in the mean, where the best estimate available of the process mean is the current forecast.

3. Levels Setting

The UICP system requires the computation of the reorder point (R) and reorder quantity (Q) for each quarter. These "levels" are determined in the simulation by the routine *LoadLevels*, which makes use of a compiled version of PD-82, UICP's level setting program. System parameters are specified by the user while current

forecast and mark code data are provided by the simulation. A description of the model used by PD-82 is provided in Chapter II.

The modified Silver model computes a potentially different reorder point and reorder quantity for each supply demand review. A description of the model and how it determines these values was provided in Chapter III.

4. Supply Demand Reviews

The term supply demand review (SDR) as used here should not be interpreted to represent the full range of functions covered by UICP's SDR process [Ref. 21: Chapter 3]. The simulation routine SDR does model the fundamental procedure of UICP's SDR process; i.e., comparing current assets (inventory position) to forecasted requirements and making a proper ordering decision. In the UICP simulation model, inventory requirements are specified by the reorder point (R). In the modified Silver model, inventory requirements are determined in a sub-procedure called *SilverModel.* In each case, inventory position is computed as the current on hand assets minus the backordered quantity plus the quantity on order.

In addition to asset and requirements comparison, the *SDR* routine contains the basic timing routine, receipt and demand processing routines, and a majority of the statistics of interest. As indicated earlier, the simulation uses a fixed-increment time advance clock with a Δt time interval of one week. At the end of each week receipts are processed first. Backorders, if any, are then filled on a first-in, first-out policy. Time-weighted units short (TWUS) is collected on a weekly basis and later converted to a daily basis to compute customer wait time statistics. Demands are then processed

and the inventory position is adjusted accordingly. Next the current assets to requirement review is conducted to determine if a reorder is required. If so, a reorder is generated with a randomly generated lead-time. Since outstanding orders are maintained in a priority queue by due-in date, order cross-over is possible.

Since the actual distribution of lead times is unknown, it is assumed that lead-times are approximately Normal about their mean. The default variance is the same that is used in the UICP model when computing lead time demand; i.e., 1.57*mean lead time. Since lead times of less than two quarters or more than three or four years are not realistic [Ref. 23], the generated lead times are truncated at two and fourteen quarters.

All statistics collected in the *SDR* routine are only for the steady state statistics collection period specified at the start of the simulation run. These statistics include average customer wait time (ACWT) in days, average customer wait time for backordered requisitions (ACWTBO) in days, and the percentage of requisitions filled, called supply material availability (SMA). Since it is assumed that each requisition is for a single unit, the formulas for each of these measures of effectiveness (MOE's) are as follows:

$$ACWT = \frac{TWUS}{Total\ Demand}$$

$$ACWTBO = \frac{TWUS}{Backorders\ Filled}$$

and,

$$SMA = 1 - \frac{Backorders\ Total}{Total\ Demand}$$

where TWUS is given in days. These MOE's are computed on both a quarterly and a cumulative basis.

Other statistics collected in the *SDR* routine are total cost, investment level and inapplicable assets. Total cost is defined to be the sum of material, administrative ordering, holding, and shortage costs. This cost is computed at the end of each replication for the entire steady state collection period. The average quarterly investment level is defined as the average on-hand plus on-order quantity (in units) for a given quarter. On-order assets are included in this computation because funding is obligated at the time of order. A cumulative average quarterly investment level is computed for each replication, representing the average investment level over all quarters. In this study, inapplicable or excess assets are defined to be any quantity in excess of two years worth of demand. In the UICP simulation model where demand is assumed to be constant, the current quarterly forecast is used to calculate the two-year quantity. In the modified Silver model, the extended forecast is used in the estimation of excess inventory. If the two-year projection period exceeds the forecast horizon, the last forecast of the forecast horizon is used for each of the remaining periods in calculating the two-year quantity.

D. INITIALIZATION AND TERMINATION

1. Number of Replications

Simulations are computer-based statistical experiments. Thus, if results are to have any meaning, appropriate statistical techniques must be applied to the design and analysis of the experiments. Estimating the behavior of a model from a single simulation replication could lead to highly erroneous results if the variance of the underlying process is large. That is, each replication is only a realization of a random variable and an appropriate sized sample of such realizations must therefore be selected in order to make any reasonable statistical inference regarding the model.

The selection of a sample size for the experiments presented in this thesis was based on two considerations. First, in order to conduct a wide range of experiments (92 scenarios for each model), consideration had to be given to computer run time. In this case, the run time for a single replication of 115 quarters of the UICP simulation is approximately 1.5 minutes on an IBM compatible 486-33 MHZ personal computer (times vary slightly based on system configuration). This translates into an approximate run time of 2.5 hours for every 100 replications. Since access to multiple computers was only available at night, a total run time not exceeding the available time was preferred. Secondly, it was desired that the number of replications provide a reasonable measure of statistical significance with as little probability of making a type II error as possible. Therefore, since protection against Type II errors only increases with sample size, a maximum sample size was chosen for the time

available. This turns out to be 500 replications which corresponds to a run time of approximately 12.5 hours for replications that are 115 quarters long.

There are several measures of the measurement error associated with a given sample size. One measure, absolute error, is defined as the absolute value of the difference of the estimated mean value and the population mean. Absolute error can be used to determine the number of replications for a given level of significance α by finding the number of replications which yield a confidence interval half length that is less than or equal to the preselected absolute error value [Ref. 29: p. 536-537].

Alternatively, an attained absolute error level can be obtained for a given α and a given number of replications. These attained values can then be further viewed in terms of practical significance. Table 4.1 provides the mean of three effectiveness measures for four UICP simulation runs of 500 replications each. In each case demand was generated from a Normal distribution with a stationary mean. System parameters were the same for all four runs with the exception of unit price ($250 and $100, respectively, for $\mu=4$ and $\mu=12$), mean demand, and the variance of demand. The results displayed include the mean value and the 95%Normal confidence interval limits.

An attained absolute error value can easily be obtained from the data in Table 4.1. These values are given in Table 4.2 below. For example, to interpret a table value consider ACWT for the case where the mean is 12 and the variance is 23. Based on the results of 500 replications, 95% of the time we would expect a replication mean to have an absolute error of at most 0.21 days (1.35-1.14), and 5%

TABLE 4.1. MEAN AND CONFIDENCE INTERVAL LIMITS.

Measure of Effectiveness	$\mu=4$ $\sigma^2=2.6$	$\mu=4$ $\sigma^2=31.4$	$\mu=12$ $\sigma^2=23$	$\mu=12$ $\sigma^2=282$
ACWT (Days)	1.08 0.86 (L) 1.29 (U)	8.60 7.35 (L) 9.86 (U)	1.14 0.92 (L) 1.35 (U)	8.4 7.23 (L) 9.60 (U)
Avg. Qtrly. Investment (Units)	69.80 69.39 (L) 70.22 (U)	102.43 101.0 (L) 103.85 (U)	206.30 205.23 (L) 207.36 (U)	294.18 290.37 (L) 297.98 (U)
Total Cost (Dollars)	151,631 150,856 (L) 152,406 (U)	211,000 208,643 (L) 213,356 (U)	180,549 179,534 (L) 181,563 (U)	259,396 255,237 (L) 263,555 (U)

TABLE 4.2. ABSOLUTE ERROR VALUES.

Measure of Effectiveness	$\mu=4$ $\sigma^2=2.6$	$\mu=4$ $\sigma^2=31.4$	$\mu=12$ $\sigma^2=23$	$\mu=12$ $\sigma^2=282$
ACWT (Days)	0.21	1.26	0.21	1.2
Avg. Qtrly. Invest. (Units)	0.42	1.42	1.06	3.8
Total Cost (Dollars)	775	2356	1014	4159

of the time we would expect the absolute error to exceed 0.21 days. Note that while the absolute error of total cost appears to be rather large ($1014), it is considerably less than 1% of the mean value.

2. Seed Selection

As previously discussed, the simulation models can generate up to 20,000 unique demand streams each of length 100,000. During the set-up phase of a simulation run the user may select a starting seed from any one of the 20,000 seeds, less the number of replications selected. The starting seed and all subsequent seeds

are generated automatically by the program. Alternatively, the user may choose to manually input up to 100 individual seed values.

Seed selection for a simulation run is directly related to experimental design and output analysis. When running a single model it is usually desired that each replication is an independent realization of the random process, thus allowing the experimenter to apply relatively simple data analysis techniques to the results. Statistical "independence" results from selecting different seeds for each replication. When comparing the performance of two different models though, independence may not be as desirable. Although independent data allows for many direct statistical comparison techniques, comparative model performance using exactly the same data may be most preferable from many viewpoints. In the latter case, an analysis technique for dependent samples would be employed for statistical comparisons.

For this study, independent demand streams were generated for each simulation run or scenario using the first 500 seeds. Corresponding runs for each model where made with identical demand streams[11]. Since model comparisons are based on dependent samples, a paired-t test is employed to compare performance measures [Ref. 32: p. 572-575].

[11]To ensure demand streams for each model are identical, all demands observations are generated at the start of a replication. This is necessary since procurement lead times are generated from the same random generator and different reorder distributions would corrupt the demand observation stream.

3. System Parameters

For the purpose of this study, the system parameter values shown below are the default input parameter values and remain constant from quarter to quarter:

- The probability break point is 0. This code is used in UICP in the determination of the probability distribution of lead time demand. In this case, lead time demand is assumed to have a Normal distribution unless it is a very low demand item (Mark Code 0), where it is assumed to have a Poisson distribution.*
- The shelf life code is 0. This code is used in UICP to set the reorder level and order quantity shelf life constraints for an item. In this case, the shelf life constraints are disregarded.*
- The requisition size is fixed at one unit.
- The annual obsolescence rate is 0.12.
- The annual storage rate is 0.01.
- The annual time preference rate is 0.10.
- The minimum risk constraint is 0.10.*
- The maximum risk constraint is 0.35.*
- The low limit for the reorder point is one unit.
- The shortage cost is $1000.00 per unit-year short.
- The administrative cost of placing an order is $850.00

With the exception of requisition size, the default values given are representative of those used by SPCC for many consumable items [Ref. 33]. The parameters marked with an asterisk (*) are not used in the modified Silver model, although a specified risk level is an input parameter in the Modified Silver model. Its determination is discussed in more detail in the next chapter. The obsolescence, storage, and time preference rates are expressed as a fraction of unit cost per year; i.e., the cost to hold one dollar's worth of material in inventory for one year. Unit price, mean demand, and the variance of demand vary with each scenario. Except when otherwise noted, mean procurement lead time is set to 8 quarters and the variance of lead time is equal to 1.57 times the mean.

4. Initial Conditions

Inherent to stochastic simulations is the initial transient or start-up problem; i.e., performance measures for a terminating simulation depend explicitly on the initial state of the system. A terminating simulation is one for which there is a pre-specified event or occurrence that determines the length of each run or replication. In our case, the event is the ending quarter specified by the user during the simulation run setup. The technique most often used to deal with the initial transient problem is called "warming up the model" or "initial-data deletion" [Ref. 29: p. 545]. Using this technique, data is discarded or simply not collected for the random variables being measured until transient means converge to the steady state mean. This technique is employed in this study. A disadvantage of this approach is that a sizable portion of the simulated data is discarded. This can be partially compensated for by setting initial conditions as close as possible to either theoretic or expected steady state conditions in order to accelerate convergence.

Although several techniques exist for determining the length of the warm-up period, the method selected in this study is a rather simple, graphical procedure attributable to B.L. Welch [Ref: 29: p. 545-546]. The technique involves graphing a moving average of the results of n independent replications of a simulation and determining the point at which the transient mean curve "flattens" out. This point is the end of the warm-up period.

For the purpose of this study, the random variable selected for initial transient analysis was quarterly investment, defined as the average quantity on hand

plus on order. Investment level was chosen since it is a direct function of two of the principal simulation state variables. As indicated previously, the warm up period can be reduced by the selection of appropriate initial starting conditions. For the UICP simulation model, the initial on hand inventory was set to the expected steady state on hand quantity for the EOQ model. This quantity is defined as one half of the initial reorder quantity plus initial safety stock, where safety stock is equal to the reorder point minus lead time demand [Ref. 29: p. 275]. The number of outstanding orders at the start of the simulation is set equal to the integer value of the mean lead time demand divided by the reorder quantity, rounded down. The total number of outstanding orders is the product of this integer value and the reorder quantity [Ref. 4: p. 32-33]. These initial outstanding orders are then scheduled to arrive at equal intervals over an initial period that is the length of a mean lead time period. The same initial conditions were manually entered for corresponding runs of the modified Silver model.

Since this study involved running many demand profile scenarios, it was not practical to analyze the warm-up period for each and every case. Rather, a sample of scenarios was analyzed to determine a single, conservative starting point to be applied to all scenarios. Seven stationary demand scenarios of 100 replications each were analyzed for each model using a 20-quarter moving average window (W=20). Based on this analysis, it was determined that a warm-up period of 25 quarters was sufficient. The fourteen Welch procedure graphs are contained in Appendix B.

5. Terminating Conditions

Both the UICP and modified Silver simulation models are terminating simulations. Because of this, several assumptions and adjustments need to be made to ensure the statistics are not affected significantly by the termination event.

First, time-weighted units short (TWUS) is measured from the time a unit is placed in a backorder status to the time it is actually filled. If a replication terminates with units in a backorder status, it is assumed that all outstanding backordered units will be filled by the next stock reorder due in.[12] This allows for the collection of TWUS for all backordered units.

Secondly, although the modified Silver model adjusts well to a final forecast horizon, the UICP model assumes steady state conditions and will continue to do so up to the final quarter. The modified Silver model satisfies requirements for a specified forecast horizon. Since no demand is forecast past the last quarter, the forecast horizon is incrementally reduced as the final quarter is approached. No orders will be generated during the final mean lead time period since there are no future requirements for an order to meet. Since the UICP model has no such stopping mechanism, the user can specify the last quarter for which statistics are to be collected in each simulation model. Although in this study where the focus is on declining demand and there are generally excess assets on hand, reducing the likelihood for additional orders near the end of a replication, the final statistics collection quarter for

---

[12]This procedure may lead to a slight underestimation of TWUS for any units backordered at the end of a replication. In our case, where a majority of the cases studied involve declining demand, there are generally few backorders at the end of a replication.

each model was set at minus ten quarters from the ending quarter. In our case stopping at minus eight quarters from the ending quarter would be sufficient since the modified Silver model stops ordering at minus a mean lead time from the ending quarter, but ten quarters was more convenient for setting up run specifications.

V. RESEARCH METHODOLOGY AND RESULTS

A. OVERVIEW

Simulation experiments are a special case of experiments which, in general, afford more control over inputs or factors than can usually be achieved in a physical experiment with a system. In simulation modeling, experimental design is used to decide which configurations to simulate so that the desired information can be obtained with the least amount of simulating [Ref. 29: p. 657]. Since this research is directed towards the comparative performance of two simulation models, each with numerous parameters, the appropriate selection of factors is even more critical.

In designing the experimental settings for this study, demand profile, demand distribution, unit price, and the distribution of replenishment lead times were selected as experimental factors. Comparison of model performance is based on the following output performance measures or responses:

- Average customer wait time (ACWT);
- Average customer wait time for backordered requisitions (ACWTBO);
- Supply material availability (SMA);
- Average quarterly investment level;
- Total cost (for the steady state collection period);
- Ending excess assets.

Because of the number of factors and the range of possible levels, a total of 92 experiments or scenarios were evaluated. A listing of the scenarios is contained in Appendix C.

B. EXPERIMENTAL FACTORS

1. Demand Profile

For the purpose of this study, the various demand data streams or "profiles" are classified as stationary, cyclic, declining or increasing. The cyclic demand profile includes a period of initial increase corresponding to a build-up period followed by a stationary period and then a declining period. Demand profiles are further categorized by the character of the trend; i.e., linear, step or exponential, and the length of the trend period. Step trend periods have a step change in the mean demand followed by a stationary period, then another step change in the mean demand followed by a stationary period, and so on, until the trend period is over. Exponential trend periods are either concave upwards or concave downwards. Concave upward trends have an initial slow trend rate followed by a higher trend rate. Concave downward trends have an initial high trend rate followed by a slower trend rate. Concave trend patterns are displayed graphically in Figure 5.1. The minimum trend period length is eight quarters and the maximum trend period length is twenty quarters. Representative realizations of the sixteen demand profiles used in this study are numbered and graphically displayed in Appendix D. With the exception of the mean level of demand, the same demand profile characteristics are used for both Normally and Poisson generated demand. The profile number is cross referenced to scenario under the profile heading in the scenario listing in Appendix C. In all cases the models are allowed to reach steady state conditions prior to implementation of any non-stationary mean condition.

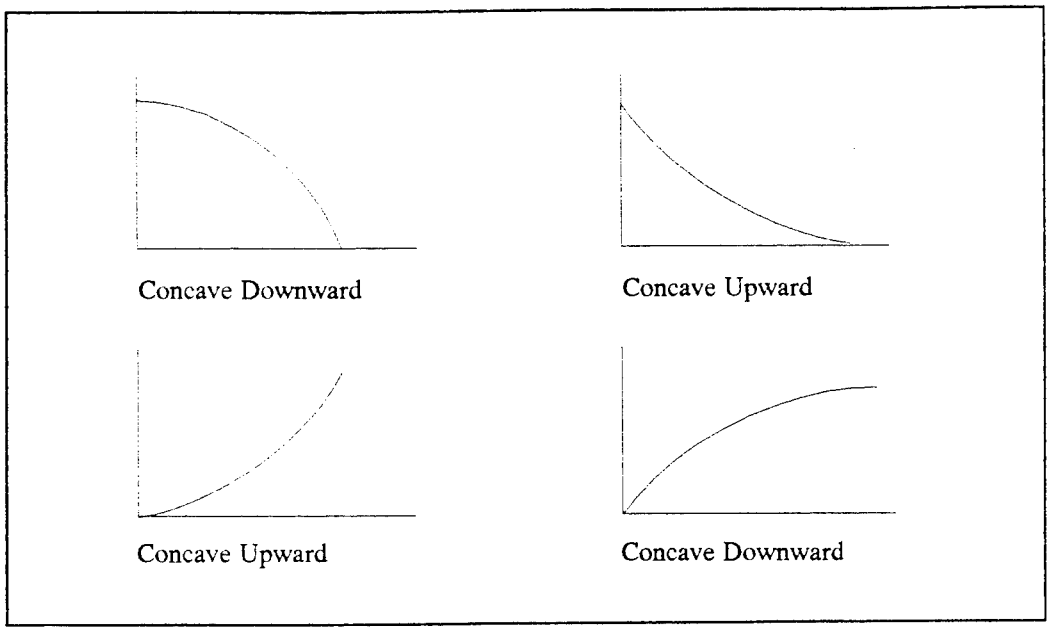


FIGURE 5.1. Exponential Trend Patterns.

2. Demand and Lead Time Distribution

In stochastic inventory systems, the distributions of both demand and replenishment lead time are critical elements in determining system behavior. First, theoretical model assumptions and parameters are usually based on some assumed demand and lead time probability distribution. Extreme variation from these assumed distributions may significantly affect model performance. Secondly, safety stock is a direct function of the variability of demand and lead time. Excessive safety stock can be costly and insufficient safety stock can reduce customer support.

In this study, demands are generated from a Poisson or a Normal probability distribution. Mean demand levels are categorized as very low (μ=0.25 or 1.0), low (μ=4), moderate (μ=12) or high (μ=25). For Normally generated

demand, variability levels are categorized as low ($\sigma/\mu=0.4$), moderate ($\sigma/\mu=0.8$), or high ($\sigma/\mu=1.4$).

The distribution parameters for replenishment lead time are also varied. The default mean lead time is 8 quarters with a variance of 12.56 (1.57 times the mean). Model performance is also evaluated under two other alternatives, one where mean lead time is reduced to 4 quarters and a second where lead time is fixed at 8 quarters (deterministic case).

3. Unit Price

As discussed in Chapter III, the approximate length of an order cycle for the modified Silver model is a function of unit price, mean demand, holding and ordering cost rates. Since holding and ordering cost rates are fixed in this case, the length of an order cycle is simply a function of mean demand and unit price. Therefore, to compare model performance with regards to order cycle length, unit price is varied for different levels of mean demand. For the modified Silver model, a low dollar value level yields longer order cycle lengths (4-6 periods), while a higher dollar value yields shorter order cycle lengths (1-3 periods).

4. Other Parameters

a. Risk

In the UICP model a constrained risk value is computed each quarter by the model. In the modified Silver model the risk value is an input parameter and remains constant from quarter to quarter. In comparing the two models, the selection

of an appropriate risk value for the modified Silver model is important because of the dependent relationship between risk, investment level and system effectiveness. This dependence leads to two approaches in selecting a risk value. The first is to select risk to meet a constraint for some performance measure such as ACWT or SMA, allowing the investment level to be determined by the model. The second is to select risk to meet some specified investment level, allowing the performance measures to be determined by the model. In this study, the latter approach is employed; i.e., the risk value is selected such that the average investment level is approximately equal to the attained average investment level of the corresponding UICP simulation run. Since investment levels vary significantly with demand that is trending, the investment levels for non-stationary demand scenarios were based on stationary demand test scenarios with the same initial demand distribution parameters.

b. Buffer

Unique to the modified Silver model is a buffer quantity, expressed as the product of a coefficient (denoted as b) and the standard deviation of forecast error over the interval associated with forecasted demand X2, from the present review, t_0, until time T-1. As indicated in Chapter III, the amount of buffer, if any, should be a management decision involving the trade-off between investment level and the number of replenishments. No attempt is made in this study to optimize the selection of the coefficient (b) for each scenario. Rather, a series of simulation runs were made to determine the range of coefficient values that would keep the penalty small in terms of total cost. The test was conducted for ten stationary demand

scenarios. Experimental factors included the distribution of both demand and replenishment lead time and unit price (denoted as C). Unit price was selected as a factor to compare model performance with regards to order cycle length. Statistics were collected for 80 quarters. For each scenario, independent simulations of 645 replications were run for buffer coefficient values from 0.0 to 3.0 in 0.1 increments. Thus, for each scenario a total of 31 x 645 = 19,995 replications were run. Mean total cost values were then analyzed to determine ranges of coefficient values that were statistically equivalent based on a standard analysis of variance test. The null hypothesis stated that all means within the selected coefficient range were equal, while the alternative hypothesis stated that at least one mean was not equal. Table 5.1 summarizes the test results for a selection of ranges. A table entry of E indicates statistical acceptance of the null hypothesis at a 0.05 significance level. The general finding is that for small values of the buffer coefficient (0.0 - 0.6), there is little penalty in terms of overall cost. It should be noted that even in those cases of Poisson demand where statistical equivalence was not attained, the difference of the highest and lowest cost was less than two percent of the lowest cost value.

The trade-off to increased investment for a larger buffer value is a reduction in the number of replenishment orders. For the ten test scenarios, Table 5.2 provides the mean number of total orders for selected values of the buffer coefficient. In all cases, even a relatively small change in the buffer coefficient can have an appreciable effect upon the total number of orders. Although in this simulation study there is no constraint on total procurement workload, such constraints do exist in real

TABLE 5.1. COMPARISON TEST ON MEAN TOTAL COST.

SCENARIO	COEFFICIENT (b) RANGE					
	0.0 - 0.6	0.0 - 0.9	0.0 - 1.5	0.0 - 1.9	0.0 - 2.5	0.0- 3.0
μ=1 (Poisson), Lead Time =8 qtrs, C=1000	-	-	-	-	-	-
μ=1 (Poisson), Lead Time =8 qtrs, C=5000	-	-	-	-	-	-
μ=12, σ^2=23, Lead Time =8 qtrs, C=100	E	-	-	-	-	-
μ=12, σ^2=282, Lead Time =8 qtrs, C=100	E	E	E	-	-	-
μ=12, σ^2=23, Lead Time =8 qtrs, C=450	E	E	E	-	-	-
μ=12,σ^2=282, Lead Time =8 qtrs, C=450	E	E	E	-	-	-
μ=12, σ^2=23, Lead Time =4 qtrs, C=100	E	-	-	-	-	-
μ=12, σ^2=282, Lead Time =4 qtrs, C=100	E	E	E	-	-	-
μ=12, σ^2=23, Lead Time =8 qtrs (Fixed), C=100	E	-	-	-	-	-
μ=12, σ^2=282, Lead Time =8 qtrs (Fixed), C=100	E	E	E	E	E	E

TABLE 5.2. MEAN NUMBER OF REPLENISHMENT ORDERS.

SCENARIO	MEAN ORDER COUNT						
Coefficient (b)	0.0	0.5	1.0	1.5	2.0	2.5	3.0
μ=1 (Poisson), Lead Time =8 qtrs, C=1000	12.5	10.1	8.5	7.3	6.3	5.7	5.1
μ=1 (Poisson), Lead Time =8 qtrs, C=5000	31.6	22.8	18.1	15.1	13.1	11.5	10.1
μ=12, σ^2=23, Lead Time =8 qtrs, C=100	18.0	15.4	13.6	12.0	10.8	9.9	9.0
μ=12, σ^2=282, Lead Time =8 qtrs, C=100	15.1	12.6	10.8	9.6	8.3	7.7	7.1
μ=12, σ^2=23, Lead Time =8 qtrs, C=450	54.8	43.1	35.3	30.2	26.5	23.5	21.0
μ=12,σ^2=282, Lead Time =8 qtrs, C=450	38.2	29.0	23.3	19.8	16.7	15.4	13.7
μ=12, σ^2=23, Lead Time =4 qtrs, C=100	18.9	16.1	14.2	12.4	11.3	10.2	9.3
μ=12, σ^2=282, Lead Time =4 qtrs, C=100	18.1	15.0	12.6	11.1	9.6	8.8	7.6
μ=12, σ^2=23, Lead Time =8 qtrs (Fixed), C=100	18.2	15.6	13.7	12.1	10.9	9.9	9.1
μ=12, σ^2=282, Lead Time =8 qtrs (Fixed), C=100	16.5	13.7	11.6	10.2	8.9	8.3	7.6

life. Therefore, imposing a management decision in the study to restrict total orders would be reasonable. With this is mind, a fixed buffer coefficient value of 0.5 was selected because it seems to provide a reasonable balance between investment level and the total number of orders per year.

c. *Maximum Order Cycle Length During Decline*

To comply with the DOD maximum order quantity constraint, the default setting for the maximum order cycle length for the modified Silver model is 6 quarters. An additional input parameter allows the user to further tighten this constraint during periods of declining demand. Although such a constraint is not necessary from a theoretical viewpoint, it does have some practical appeal from a management perspective.

When a period of declining demand is first forecasted (i.e., a declining demand pattern is detected within the forecast horizon), it is possible for the modified Silver model to increase the investment level for several quarters prior to actually reducing levels in response to the decline. This occurs in cases where the order cycle length has normally been less than the maximum length, and because of the declining forecast, the model's heuristic determines that a longer order cycle is more optimal. If the decline is gradual, then the order quantity and thus investment level may be greater. This creates a temporary increase or "hump" in investment at the start of a declining period. Although the longer order cycle has a lower total cost, there may be a genuine concern about increasing the average investment level as you are about to enter a period of declining demand. Thus, given this concern and the general uncertainty of mean demand information during non-stationary periods, the user is allowed to restrict the maximum order cycle length to less than 6 quarters just before or during periods of declining demand. This parameter has been set to a default value of 4 quarters for all simulation runs.

In some cases, a similar phenomena occurs at the end of the decline period. In these cases, as the model detects the impending return to stationarity of the mean demand, the maximum order cycle constraint for the declining demand is removed. Since demand is at a lower level, the order cycle length may increase significantly (i.e., from 4 to 6 quarters), causing a temporary increase or "hump" in investment. Since this new order cycle length has a lower cost and we've reached a new period of stationary demand, the model is allowed to perform under its normal assumptions. In the case where demand declines to zero, this phenomenon does not occur.

C. SIMULATION RESULTS

1. Stationary Demand

The first simulation series examines the effects of mean demand, demand variability, procurement lead time, and unit price on model performance. The series consist of thirty-two simulation runs (see Appendix C, experiments 1 through 32). Steady state statistics are collected for 80 quarters. In each case, mean demand is stationary during all 80 quarters. Model comparison is based on a paired t-test of the difference of the means (sample size=500) for cumulative measures of effectiveness[13].

[13]Cumulative measures of effectiveness, as opposed to quarterly, represent aggregate performance. For instance, cumulative ACWT at any point in time is defined to be total TWUS up to that point in time divided by the total demand up to that point in time. Similarly, cumulative SMA represents the percentage of total requisitions that have been satisfied when submitted (i.e., the percentage of total requisitions not backordered when submitted up to that point in time).

Table 5.3 provides a summary of the results for Normally generated demand. The table columns correspond to the measures of effectiveness identified in the bottom row of the table. The first five columns have measures which were defined in Chapter IV, Section C. The sixth column, "Excess," refers to the amount of ending excess inventory. The table indicates which model performed better for each effectiveness measure. An "S" signifies the modified Silver model and a "U" signifies the UICP model. A dash indicates that the mean values were statistically equivalent at a 0.01 significance level.

The results clearly indicate that in most low to high demand cases, the modified Silver model will outperform or perform equally as well as the UICP model when the mean demand is stationary. Performance improves slightly with a higher demand rate or a lower unit cost. In the two scenarios where the UICP model statistically outperformed the modified Silver model in one or two single measures, the practical differences were relatively small, as is the case in many of the comparisons. This results from a large sample size and the resulting power of the statistical test.

The mean values, differences and p-values of all performance measures are listed by experiment number in Appendix E (Table E-1). This data provides additional insight into the comparative performance of the models not readily apparent in the Table 5.3. For example, by comparing the mean values for each performance measure for experiments 6, 7 and 8, the reader can see that both models perform similarly with increased demand variability. Note also by comparing the total number of orders in the last column of Table E-1, that the number of orders is significantly

higher in many cases for the modified Silver model when the unit cost is high (see experiments 13 through 22). In this study, since administrative ordering costs are included in total cost, model performance is not measured separately in terms of the total number of orders.

Because cumulative data can sometimes be misleading, both cumulative and quarterly effectiveness data are displayed graphically for a representative stationary mean demand scenario (experiment #14) in Appendix F.

TABLE 5.3. PAIRED T-TEST COMPARISON (α=0.01) FOR NORMAL DEMAND WITH STATIONARY MEAN.

Exp #	Mean	Variance	Low Dollar Value						High Dollar Value					
			1	2	3	4	5	6	1	2	3	4	5	6
3, 14	Low (4)	Low (2.6)	-	-	-	S	S	-	S	S	S	S	-	S
4, 15	Low (4)	Mod (10.2)	S	S	S	-	-	-	S	S	-	S	U	S
5, 16	Low (4)	High (31.4)	-	S	S	S	S	S	-	U	U	S	-	S
6, 17	Mod (12)	Low (23)	S	S	S	S	-	S	S	S	S	S	S	S
7, 18	Mod (12)	Mod (92)	S	S	S	-	S	-	S	S	S	S	S	-
8, 19	Mod (12)	High (282)	S	S	S	S	S	S	S	-	-	S	-	-
9, 20	High (25)	Low (100)	S	S	S	S	S	-	S	S	S	S	S	S
10,21	High (25)	Mod (400)	S	S	S	S	S	-	S	S	S	S	-	-
11,22	High (25)	High (1225)	S	S	S	S	S	-	S	S	-	S	-	-

Col. 1=ACWTBO Col. 2=ACWT Col. 3=SMA Col. 4=Investment Col. 5=Total Cost Col.6=Excess

Table 5.4 provides a similar summary of the results for Poisson generated demand. In this case the results are less conclusive than those noted above. One hypothesis to explain this inconsistency is that the assumption made in the modified Silver model that forecast variability can be measured using MAD is especially bad for very low levels of demand (see Chapter III). To test this hypothesis, a series of additional tests were conducted for the Poisson demand scenarios where the variances

of forecasted demands X1 and X3 for the modified Silver model were calculated using the same power rule that is used by SPCC in the UICP model's computation of the variance of lead time demand for very low demand items. This power rule was derived by SPCC using regression analysis and is expressed as: lead time demand variance $=3.869$ (lead time demand) $^{1.378}$. In the modified Silver case, demand X1 or X3, covering an interval of length $L+1$, is used instead of lead time demand. Table 5.5 gives the results for these additional tests. The results indicate that MAD should not be used with very low mean demand levels. It should also be noted that the use of the same coefficient and exponent in the modified Silver model does not imply that these values are correct. Rather, it is meant to suggest that a similar power rule computation might be developed for the modified Silver model for use with very low demand items.

TABLE 5.4. PAIRED T-TEST COMPARISON (α=0.01) FOR POISSON DEMAND WITH STATIONARY MEAN.

Exp #	Mean	Variance	Low Dollar Value						High Dollar Value					
			1	2	3	4	5	6	1	2	3	4	5	6
1, 12	0.25	0.25	-	S	S	S	S	S	-	S	S	S	S	S
2, 13	1	1	U	U	U	U	U	-	-	-	-	S	-	-

Col. 1=ACWTBO Col. 2=ACWT Col. 3=SMA Col. 4=Investment Col. 5=Total Cost Col. 6=Excess

TABLE 5.5. PAIRED T-TEST COMPARISON (α=0.01) FOR POISSON DEMAND WITH STATIONARY MEAN AND POWER RULE.

Exp #	Mean	Variance	Low Dollar Value						High Dollar Value					
			1	2	3	4	5	6	1	2	3	4	5	6
1, 12	0.25	0.25	S	S	S	S	S	S	S	S	S	-	-	S
2, 13	1	1	-	-	-	S	-	S	-	-	-	S	S	S

Col. 1=ACWTBO Col. 2=ACWT Col. 3=SMA Col. 4=Investment Col. 5=Total Cost Col. 6=Excess

Table 5.6 provides a summary of the results when procurement lead time is varied. In the first series lead time is probabilistic with a mean of four quarters and a variance of 6.28 (1.57 times mean). In the second series lead time is fixed at eight quarters. The unit prices are the same as those used in the low dollar cases in the above experiments. The effects of varying lead time on the performance of the modified Silver model are consistent with the behavior of the UICP model under the same conditions.

TABLE 5.6. PAIRED T-TEST COMPARISON (α=0.01) FOR STATIONARY DEMAND WITH VARYING LEAD TIME.

Exp #	Mean	Variance	Mean Lead Time =4 Qtrs (Variable)						Lead Time=8 Qtrs (Fixed)					
			1	2	3	4	5	6	1	2	3	4	5	6
23,28	1	1 (Poisson)	U	U	U	S	-	-	U	U	-	-	-	-
24,29	Low (4)	Low (2.6)	-	-	-	-	-	S	S	S	S	S	-	-
25,30	Low (4)	High (31.4)	-	S	S	S	S	S	S	S	S	S	S	S
26,31	Mod (12)	Low (23)	S	S	S	S	-	S	S	S	S	S	-	-
27,32	Mod (12)	High (282)	S	S	S	S	S	S	S	S	S	S	S	S
Col. 1=ACWTBO Col. 2=ACWT Col. 3=SMA Col. 4=Investment Col. 5=Total Cost Col. 6=Excess														

Although results for all of the performance measures are presented in Table E-1 of Appendix E, Table 5.7 provides the reader a summary of the effects of varying lead time on mean ACWT. A close examination of the results in Table 5.7 suggests some inconsistencies with expected behavior, although the results are consistent between models. For instance, one would expect ACWT to be considerably less for the case when mean lead time is four quarters (variable) than when mean lead time is eight quarters (variable). This, however, is only true for the high variance demand cases. In the low variance demand cases ACWT increased as mean lead time

was reduced. This unexpected behavior results from a simulation assumption rather than a model assumption. Recall from Chapter IV that observed lead times are generated from a Normal distribution truncated at two and fourteen quarters. In the case where a mean of four quarters is used to generate lead times, the observed mean lead time will be greater than four, since four is near the lower truncation point. Since safety stock is based, in part, on expected lead time, safety stock will be underestimated. In the case where the expected lead time is four quarters, the simulation results for ACWT may be somewhat higher than expected. If the variance of demand is high the additional safety stock due to the variability of demand appears to offset the underestimation resulting from a higher than assumed mean lead time. In the case of a mean lead time of eight quarters (the default mean lead time), the shifting upward of the observed mean lead time will be less evident since eight is more central to both truncation points. Results similar to those shown in Table 5.7 were obtained for the other performance measures. However, the reader should note that the relative investment levels between scenarios (see Appendix E) are consistent with expected behavior.

TABLE 5.7. MEAN ACWT (IN DAYS) FOR STATIONARY DEMAND WITH VARYING LEAD TIMES.

Mean	Variance	UICP			MODIFIED SILVER		
		LT=8 Var	LT=4 Var	LT=8 Fix	LT=8 Var	LT=4 Var	LT=8 Fix
1	1 (Poisson)	4.39	5.45	2.60	6.82	6.99	4.72
Low (4)	Low (2.6)	1.08	1.33	0.78	1.0	1.10	0.48
Low (4)	High (31.4)	8.60	6.39	11.91	7.25	5.20	10.43
Mod (12)	Low (23)	1.14	1.33	0.83	0.72	0.87	0.48
Mod (12)	High (282)	8.42	6.43	13.94	6.04	4.36	10.13

In this section it has been shown that the modified Silver model's performance is equal to or better than the UICP model for most stationary mean demand scenarios. The primary exception is for the Poisson demand case. In this case, though, it has been shown that modifying the model assumption of using MAD to estimate forecast variability will lead to significant improvements. It has also been shown that both models behave similarly when the distribution of procurement lead time is varied.

2. Cyclic Demand

The next simulation series involves eight cyclic demand scenarios (see Appendix C, experiments 33 through 40) and two generated demand profiles (see Appendix D, profiles 2 and 3). In the case of Poisson demand, the mean is initially set at 0.25 units per quarter, increases to 2 units per quarter, and declines back to a mean of 0.25 units per quarter. For Normal demand, the mean is initially set at 4 units per quarter, increases to 32 units per quarter, and declines back to a mean of 4 units per quarter. In each case, the trends are exponential. Trend periods are further characterized by their length. A steep or short trend period is 8 quarters in length and a slow or long trend period is 20 quarters in length. The trend starting (T/S Qtr) and ending quarters (T/E Qtr), and parameters are listed by experiment number in Appendix C. Statistics are collected for a total of 85 quarters. For all experiments, the model comparisons are based on a paired t-test on the differences of the means (sample size=500).

Table 5.8 provides a summary of the results for all 8 scenarios. Table 5.9 provides the mean difference (Modified Silver - UICP) and percentage change in ACWT, total costs and ending excess for each scenario. Again, all measures are as defined in Chapter IV, Section C. The mean values, differences and p-values of all performance measures are listed by experiment number in Appendix E (Table E-2). Cumulative and monthly effectiveness data are displayed graphically in Appendix G for a representative cyclic demand scenario (experiment #35).

TABLE 5.8. PAIRED T-TEST COMPARISON (α=0.01) FOR CYCLIC DEMAND.

Exp #	Mean	Variance	Steep Trends (8 qtrs)						Slow Trends (20 qtrs)					
			1	2	3	4	5	6	1	2	3	4	5	6
33,37	0.25 (P)	0.25	S	S	S	S	S	S	S	S	S	S	S	S
34,38	4 (N)	Low (2.6)	S	S	S	S	S	S	S	S	S	S	S	S
35,39	4 (N)	Mod (10.2)	S	S	S	S	S	S	S	S	S	S	S	S
36,40	4 (N)	High (31.4)	S	S	S	S	S	S	S	S	S	S	S	S
Col. 1=ACWTBO Col. 2=ACWT Col. 3=SMA Col. 4=Investment Col. 5=Total Cost Col. 6=Excess														

TABLE 5.9. MEAN DIFFERENCE AND PERCENT CHANGE IN ACWT, TOTAL COSTS AND ENDING EXCESS FOR CYCLIC DEMAND.

Exp #	Mean	Variance	Steep Trend (8 qtrs)			Slow Trend (20 qtrs)		
			ACWT (Days)	Total Costs ($)	Excess (Units)	ACWT (Days)	Total Costs ($)	Excess (Units)
33,37	0.25 (P)	0.25	-18.91 (-36.9%)	-20,213 (-7.8%)	-6.70 (-43.2%)	-29.20 (-53.3%)	-18,581 (-6.8%)	-9.90 (-48.0%)
34,38	4 (N)	Low (2.6)	-24.16 (-77.8%)	-121,176 (-27.1%)	-142.24 (-54.2%)	-34.15 (-90.6%)	-171,830 (-33.4%)	-231.67 (-59.0%)
35,39	4 (N)	Mod (10.2)	-27.88 (-63.7%)	-141,981 (-26.2%)	-150.76 (-44.9%)	-40.89 (-79.7%)	-200,816 (-32.6%)	-209.66 (-46.2%)
36,40	4 (N)	High (31.4)	-30.80 (-51.2%)	-180,483 (-25.1%)	-175.92 (-39.8%)	-48.09 (-71.8%)	-256,938 (-32.1%)	-203.94 (-37.5%)

The data indicates that the modified Silver model performs significantly better than the UICP model when mean demand is non-stationary. This is evident even in the case of Poisson demand. It should be noted that, in reality, planned program requirements would be the reason for some anticipated increases in demand. Planned program requirements are non-recurring requirements for material that cannot be forecasted by the UICP system using past demand observations [Ref. 21]. Funding these requirements would partially offset the large differences in performance measures realized between the UICP model and the modified Silver model..

In addition to running the modified Silver model with variable forecasts, eight simulation runs were performed with an unmodified or "fixed" forecast (i.e., future forecasts are assumed to be the same as the present forecast). The purpose of these runs is to compare model performance with the same forecast process and assumptions used by the UICP model. Table 5.10 provides a summary of the results for these additional runs. These runs are identified by the same experiment number suffixed with an "F." A complete set of results are contained in Appendix E (Table E-5). The results are consistent with previous results with stationary mean demand; the modified Silver model will perform equal to or better than the UICP model given the same, stationary forecast information, with the possible exception of very low demand items.

TABLE 5.10. PAIRED T-TEST COMPARISON (α=.01) FOR CYCLIC DEMAND AND MODIFIED SILVER WITH STATIONARY MEAN ASSUMPTION.

Exp #	Mean	Variance	Steep Trends (8 qtrs)						Slow Trends (20 qtrs)					
			1	2	3	4	5	6	1	2	3	4	5	6
33F,37F	0.25 (P)	0.25	-	-	U	S	-	-	-	-	-	-	U	S
34F,38F	4 (N)	Low (2.6)	S	S	S	S	S	S	S	S	S	S	S	S
35F,39F	4 (N)	Mod (10.2)	S	S	S	S	S	S	S	S	S	S	S	S
36F,40F	4 (N)	High (31.4)	S	S	S	S	S	S	S	S	S	S	S	S

Col. 1=ACWTBO Col. 2=ACWT Col. 3=SMA Col. 4=Investment Col. 5=Total Cost Col. 6=Excess
(P) - Poisson Demand (N) - Normal Demand

3. Declining Demand

The third simulation series examines model performance when mean demand is declining. The series consist of thirty-six simulation runs (see Appendix C, experiments 41 through 76) and nine generated demand profiles (see Appendix D, profiles 4 through 12). For Poisson demand, the mean is initially set at 1 unit per quarter and declines to a stationary mean demand of approximately 0.25 units per quarter. For Normal demand, the mean is initially set at 25 units per quarter and declines to a stationary mean demand of approximately 6 units per quarter. An exception is made in eight cases where mean demand is made to decline to zero. Trend periods are either exponential, step or linear. The trend parameters are listed by experiment number in Appendix C. The length of the trend periods are defined in the same way as in the previous section. The steep or short trend periods begin at quarter 52 and end at quarter 59. The slow or long trend periods begin at quarter 40 and end at quarter 59. Data is collected for a total of 40 quarters (from quarter 26 to

quarter 65) which includes 6 quarters of stationary mean demand following quarter 59. For all experiments, model comparisons are based on a paired-t test on the differences of the means (sample size=500).

In a declining demand scenario the overall effectiveness of reducing investments can be measured by the impact on customer service and the ending level of excess stock. Ideally, stock levels should be reduced in such a manner that safety stock will provide an adequate buffer during the period of decline to maintain the same level of customer service that is normally achieved. In our case, given the number of scenarios, ACWT will be used as the measure of customer service.

Tables 5.11 and 5.12 provide results for the steep and slow decline scenarios, respectively. The notation is the same as that used in the previous sections, with the exception of column designations. The mean values, differences and p-values of all performance measures are listed by experiment number in Appendix E (Table E-3). As in the previous section, the data indicates that the modified Silver model performs significantly better than the UICP model when mean demand is non-stationary. In this declining demand case, the reduction in excess and total cost achieved by the modified Silver model are appreciable, with little, if any, practical impact on the level of customer service provided. This is true even in the case of Poisson demand where the underlying model assumptions may be significantly incorrect.

In interpreting the results, the following observations should be made. First, although the results often indicate a statistical difference in ACWT, in all cases

TABLE 5.11. PAIRED T-TEST COMPARISON (α=0.01) FOR DECLINING DEMAND WITH STEEP TREND.

Exp #	Mean	Var	Concave Up			Linear			Concave Down			Concave Up (0)		
			1	2	3	1	2	3	1	2	3	1	2	3
45,53,61,69	0.25 (P)	0.25	U	S	S	U	S	S	U	S	S	-	S	S
46,54,62,70	25 (N)	100	S	S	S	S	S	S	S	S	S	S	S	S
47,55,63,71	25 (N)	400	-	S	S	-	S	S	S	S	S	S	S	S
48,56,64,72	25 (N)	1225	S	S	S	S	S	S	-	S	S	S	S	S
Col. 1=ACWT Col. 2=Total Cost Col. 3= Excess (P) - Poisson Demand (N) - Normal Demand														

TABLE 5.12. PAIRED T-TEST COMPARISON (α=0.01) FOR DECLINING DEMAND WITH SLOW TREND.

Exp #	Mean	Var	Concave Up			Linear			Concave Down			Concave Up (0)			Step		
			1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
41,49,57,65,73	0.25 (P)	0.25	U	S	S	U	S	S	U	S	S	U	S	S	U	S	S
42,50,58,66,74	25 (N)	100	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S
43,51,59,67,75	25 (N)	400	-	S	S	S	S	S	S	S	S	-	S	S	-	S	S
44,52,60,68,76	25 (N)	1225	-	S	S	-	S	S	-	S	S	-	S	S	-	S	S
Col. 1=ACWT Col. 2=Total Cost Col. 3= Excess (P) - Poisson Demand (N) - Normal Demand																	

the actual difference is less than 2 days.[14] Secondly, the amount of ending excess decreases for both models as the length of the declining period increases. This is expected as a longer declining period should result in a greater portion of the excess inventory being used up. Thirdly, the performance of the modified Silver model in reducing ending excess is a function of the variability of demand. In the case of

[14] Since we would like maintain ACWT at the same level normally achieved, it is also useful to compare the results from a declining demand scenario with a stationary mean demand scenario with similar system parameters. For example, if we compare the results of the declining demand experiment #67 (ACWT=3.37) with those of the stationary demand experiment #10 (ACWT=3.25), we see virtually no change in ACWT.

Normal demand, the average percent reduction in ending excess compared to the UICP model is 53.4%, 42.1% and 35.9%, respectively for low, moderate and highly variable demand. For the Poisson demand case the average percent reduction is 33.4%. There is little variation in the percent reduction as a result of the type of decline pattern.

Table 5.13 summarizes the results for the linear declining demand cases. The mean difference is defined as the difference between the mean for the modified Silver model and the mean for the UICP model (Modified Silver - UICP). A negative value indicates that the modified Silver model value is lower. The percentage change is defined as the percent change in the mean for the modified Silver model from the UICP model. A negative percentage indicates a reduction for the modified Silver model from the UICP model. The results are similar for the other scenarios. Since there is a significant difference in the behavior of the models with the length of the declining cycle, cumulative and monthly effectiveness data are displayed graphically for both a steep trend (experiment #71) and a slow trend (experiment #59) declining demand scenario in Appendices H and I, respectively.

As in the previous section, a corresponding series of additional simulation runs were done with an unmodified forecast process. Tables 5.14 and 5.15 provide summaries for these additional runs. These "fixed" runs are identified by the same experiment number suffixed with an "F." The results are consistent with previous results using a stationary mean demand assumption. That is, with the exception of Poisson demand scenarios, the modified Silver model performs equal to or better than

TABLE 5.13. MEAN DIFFERENCE AND PERCENT CHANGE IN ACWT, TOTAL COST AND ENDING EXCESS FOR DECLINING DEMAND WITH LINEAR TREND.

Mean	Variance	Steep Trend (8 qtrs)			Slow Trend (20 qtrs)		
		ACWT (Days)	Total Cost ($)	Excess (Units)	ACWT (Days)	Total Cost ($)	Excess (Units)
1 (P)	1	1.97 (62.1%)	-5,383 (-0.65%)	-4.67 (-33.7%)	1.71 (39.1%)	-6,870 (-9.1%)	-4.16 (-34.3%)
25 (N)	Low (100)	-0.63 (-0.6%)	-18,436 (-10.7%)	-146.08 (-54.0%)	-0.68 (-66.6%)	-18,972 (-12.6%)	-97.55 (-53.4%)
25 (N)	Mod (400)	-0.11 (-2.9%)	-16,234 (-8.2%)	-129.47 (-41.9%)	-1.62 (-40.5%)	-21,643 (-12.3%)	-98.09 (-42.3%)
25 (N)	High (1225)	-1.13 (-12.8%)	-20,434 (-8.1%)	-143.11 (-37.2%)	0.06 (0.7%)	-22,741 (-9.9%)	-118.04 (-36.8%)

(P) - Poisson Demand (N) - Normal Demand

TABLE 5.14. PAIRED T-TEST COMPARISON (α=.01) FOR DECLINING DEMAND WITH STEEP TREND AND MODIFIED SILVER WITH STATIONARY MEAN ASSUMPTION.

Exp #	Mean	Var	Concave Up			Linear			Concave Down			Concave Up (0)		
			1	2	3	1	2	3	1	2	3	1	2	3
45F,53F,61F,69F	0.25 (P)	0.25	U	U	-	U	U	-	U	U	U	-	-	-
46F,54F,62F,70F	25 (N)	100	S	S	S	S	S	S	S	S	S	S	S	S
47F,55F,63F,71F	25 (N)	400	-	S	S	S	-	S	S	S	S	S	S	S
48F,56F,64F,72F	25 (N)	1225	S	S	S	S	S	S	S	S	S	S	S	S

Col. 1=ACWT Col. 2=Total Cost Col. 3=Excess (P) - Poisson Demand (N) - Normal Demand

TABLE 5.15. PAIRED T-TEST COMPARISON (α=.01) FOR DECLINING DEMAND WITH SLOW TREND AND MODIFIED SILVER WITH STATIONARY MEAN ASSUMPTION.

Exp #	Mean	Var	Concave Up			Linear			Concave Down			Concave Up (0)			Step		
			1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
41F,49F,57F,65F,73F	0.25 (P)	0.25	U	U	-	U	-	-	U	-	-	U	-	-	U	U	-
42F,50F,58F,66F,74F	25 (N)	100	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S
43F,51F,59F,67F,75F	25 (N)	400	-	S	S	S	S	S	S	S	S	-	S	S	S	S	S
44F,52F,60F,68F,76F	25 (N)	1225	-	S	S	S	S	S	-	S	S	-	S	S	-	S	S

Col. 1=ACWT Col. 2=Total Cost Col. 3=Excess (P) - Poisson Demand (N) - Normal Demand

the UICP model on all measures of effectiveness. A complete set of results are contained in Appendix E (Table E-6).

As further evidence of the ability of the modified Silver model to maintain a specified level of customer service while reducing investment levels, one can compare the results of the modified Silver model with modified forecasts to the modified Silver model with fixed forecasts. Although there is a significant difference in the performance of the model under the two forecast assumptions in terms of total cost and ending excess (see Tables E-3 and E-6), the average difference in ACWT for all scenarios is less than 0.41 days.

4. Increasing Demand

The final simulation series examines model performance when mean demand is increasing. The series consists of sixteen simulation runs (see Appendix C, experiments 77 through 92) and four demand profiles (see Appendix D, profiles 13 through 16). In the case of Poisson demand, the mean is initially set at 0.25 units per quarter and increases to a stationary mean demand of 2.0 units per quarter. For Normal demand, the mean is initially set at 4 units per quarter and increases to a stationary mean demand of 32 units per quarter. In each case, trend periods are exponential. The trend parameters are listed by experiment number in Appendix C. The lengths of the trend periods are the same as in the previous sections. The steep or short trend periods begin at quarter 52 and end at quarter 59. The slow or long trend periods begin at quarter 40 and end at quarter 59. Data is collected for a total of 65 quarters (from quarter 26 to quarter 90), allowing sufficient time for the

processes to return to a steady state condition following the increase period. For all experiments, model comparisons are based on a paired t-test on the differences of the means (sample size=500).

In an increasing demand scenario overall effectiveness can best be measured by the impact of investment levels on customer service. Investment levels must be increased to maintain a specified service level without generating excess safety levels. Again, as was the case for declining demand scenarios, model performance will be measured in terms of ACWT, total cost, and ending excess.

Table 5.16 provides results for both steep and slow trend periods. With the exception of column designations, the notation is the same as the notation that was used in the previous sections. The mean values, differences and p-values of all performance measures are listed by experiment number in Appendix E (Table E-4). As in the previous sections, the data indicates that the modified Silver model performs significantly better than the UICP model when mean demand is non-stationary.

TABLE 5.16. PAIRED T-TEST COMPARISON (α=0.01) FOR INCREASING DEMAND.

Exp #	Mean	Var	Steep Trend (8 qtrs)						Slow Trend (20 qtrs)					
			Concave Upward			Concave Down			Concave Upward			Concave Down		
			1	2	3	1	2	3	1	2	3	1	2	3
81,89,77,85	0.25 (P)	0.25	S	S	-	S	-	-	S	-	-	S	U	-
82,90,78,86	4 (N)	2.6	S	S	S	S	S	S	S	S	S	S	S	S
83,91,79,87	4 (N)	10.2	S	S	S	S	S	S	S	S	S	S	S	S
84,92,80,88	4 (N)	31.4	S	S	S	S	S	S	S	S	S	S	S	S

Col. 1=ACWT Col. 2=Total Cost Col. 3=Excess (P) - Poisson Demand (N) - Normal Demand

Table 5.17 summarizes the results for the concave upward increasing demand profile scenarios. Similar results were obtained for the concave downward scenarios. As in the previous section, since there is a difference in the behavior of the models with the length of the increasing trend period, cumulative and monthly effectiveness data are displayed graphically for both a steep trend (experiment #83) and a slow trend (experiment #79) increasing demand scenario in Appendices J and K, respectively.

TABLE 5.17. MEAN DIFFERENCE AND PERCENT CHANGE IN ACWT, TOTAL COST, AND ENDING EXCESS FOR INCREASING DEMAND.

Mean	Variance	Concave Upward Steep Trend (8 qtrs)			Concave Upward Slow Trend (20 qtrs)		
		ACWT (Days)	Total Cost ($)	Excess (Units)	ACWT (Days)	Total Cost ($)	Excess (Units)
0.25 (P)	0.25	-35.43 (-61.8%)	-1,456 (-0.9%)	0.21 (5.2%)	-16.10 (-37.6%)	165 (0.1%)	-2.19 (-4.7%)
4 (N)	Low (2.6)	-44.35 (-90.5%)	-137,538 (-35.1%)	-14.92 (-27.2%)	-26.82 (-79.5%)	-87,956 (-24.1%)	-7.66 (-14.1%)
4 (N)	Mod (10.2)	-54.47 (-81.5%)	-177,773 (-37.1%)	-9.98 (-9.5%)	-28.38 (-64.2%)	-97,573 (-22.8%)	-8.46 (-7.3%)
4 (N)	High (31.4)	-61.59 (-73.8%)	-231,835 (-34.1%)	-12.91 (-7.9%)	-32.70 (-50.6%)	-129,052 (-22.0%)	-8.97 (-4.7%)
(P) - Poisson Demand (N) - Normal Demand							

In the increasing demand case, the modified Silver model results in a lower ACWT at a lower total cost. The percent reduction in ACWT is a function of both the length of the trend period and the variability of demand. The longer the trend period the more time the forecast system has to react to the trend, resulting in a slightly better performance of the UICP model. As the variability of demand increases, the percent reduction in ACWT decreases. The overall reduction in ACWT

ranges from approximately 38% in the case of Poisson Demand to more than 95% for low varying Normal demand with a steep trend. However, as in the cyclic demand cases, one would expect some planned program requirements to be established under UICP in anticipation of increases in demand, partially offsetting the poorer performance of the UICP model.

Finally, a series of additional simulation runs were done using an unmodified forecasting process, as in the previous sections. Table 5.18 summarizes the results for these additional runs. The results are consistent with previous results using a stationary mean demand forecasting process. That is, with the exception of Poisson demand scenarios, the modified Silver model outperforms or performs equally as well as the UICP model. A complete set of results for this last series is listed in Appendix E (Table E-7).

TABLE 5.18. PAIRED T-TEST COMPARISON (α=.01) FOR INCREASING DEMAND AND MODIFIED SILVER MODEL WITH STATIONARY MEAN ASSUMPTION.

Exp #	Mean	Var	Steep Trend (8 qtrs)						Slow Trend (20 qtrs)					
			Concave Upward			Concave Down			Concave Upward			Concave Down		
			1	2	3	1	2	3	1	2	3	1	2	3
81F,89F,77F,85F	0.25 (P)	0.25	-	-	-	-	-	-	U	-	-	-	-	-
82F,90F,78F,86F	4 (N)	2.6	S	S	S	S	S	S	S	S	S	S	S	S
83F,91F,79F,87F	4 (N)	10.2	S	S	S	S	S	S	S	S	S	S	S	S
84F,92F,80F,88F	4 (N)	31.4	S	S	S	S	S	S	S	S	S	S	S	S

Col. 1=ACWT Col. 2=Total Cost Col. 3=Excess (P) - Poisson Demand (N) - Normal Demand

D. DECISION ANALYSIS

Until now, results have been presented in a simple comparative format indicating which model outperformed the other under various specific performance measures. Little direct insight has been offered as to whether the data is sufficient to justify the replacement of one model with the other. Although implementation of a new model is certainly a management decision that involves many more factors than just the measures of effectiveness outlined in this thesis, a simple approach is offered here to facilitate such a decision making process.

The problem we are faced with is commonly referred to as multiple criteria decision making (MCDM). That is, the decision maker is faced with making a decision in the presence of multiple criteria or attributes, some of which may conflict. In this case the main performance measures that are in conflict are ACWT and total cost.

One common MCDM approach is called simple additive weighting. Using this procedure, the decision maker first assigns importance weights to each attribute. Each attribute is then scaled onto some comparable measurement scale. The final scoring of an alternative is simply the sum of the product of the weights and their corresponding scaled attribute values. [Ref. 34: Chapters 1-2]

In our case, assume that a decision will be based on the following three attributes: ACWT, total cost, and ending excess. Scaling is necessary since the first attribute is measured in days, the second in dollars, and the third in units. Although several techniques exist for scaling attributes with incommensurable units, a simple

and appealing approach is to use a linear scale transformation [Ref. 34: Chapter 2]. Since our attributes are "cost" type data (i.e., the smaller the value the greater the preference) each attribute is scaled by dividing its value into the smallest corresponding attribute value from all alternatives. Therefore the smallest value will receive a scaled value of 1.0 and all others values will be less than or equal to 1.0. The appealing advantage of this technique is that it is simple and that the relative order of magnitude of each value is maintained under the transformation.

The disadvantage is that the results can be misleading if the attribute values are truly not comparable [Ref. 34: p. 101-102]. This is precisely the problem in our case. Using this technique to scale our attributes, it is possible for a relatively small difference in ACWT for small attribute values to receive the same scaled value as a relatively large difference in total cost. For example, consider an ACWT of 2 days for one model and 1.8 days for the other. The model with the lower ACWT would receive a value of 1.0 and the other would receive a value of 0.9. Consider at the same time, that total costs are 270,000 and 300,000 dollars, respectively for the two models. In this case the corresponding scaled values would again be 1.0 and 0.9. Thus, a difference of 0.2 days will receive the same relative value as a difference of $30,000 where, in reality, 0.2 days is probably less significant than $30,000.

Since we are comparing only two models, one alternative scaling procedure is to scale all attributes to a common scale based on the differences in the attribute values. For example, we might assume that differences in ACWT of less than 2 days, total cost less than 1.0% and ending excess less than 1 unit are comparable, and should

receive the same scaled value. From this baseline, we could develop a sliding scale to cover the full range of attribute values. Obviously developing such a scale would be a management function requiring much judgement and expert opinion. But, for illustrative purpose, consider the following such scaling system:

ACWT (Days)	Total Cost (Dollars)	Excess (max $\geq$ 10)	Excess (max $<$ 10)	Scale (+/-)
0-2	0-1%	0-5%	0-1	0
2-5	1-5%	5-10%	1-2	1
5-10	5-10%	10-15%	2-3	2
10-20	10-15%	15-25%	3-4	3
20-30	15-20%	25-35%	4-5	4
30-40	20-25%	35-45%	5-6	5
40-50	25-30%	45-55%	6-7	6
50-60	30-40%	55-65%	7-8	7
60-70	40-50%	65-75%	8-9	8
70-80	50-75%	75-85%	9-10	9
$>$80	$>$75%	$>$85%		10

For each range of differences in a performance measure, a corresponding scale value is assigned. For example, if the difference in ACWT is greater than 5 but less than or equal to 10, a scaled value of 2 would be assigned. In our case, if the difference favors the modified Silver model then the scale value assigned will be positive. If the UICP model performed best, the scaled value is negative. Note that the scale for ending excess has been split into a percent difference column and a unit difference column depending on whether the largest value being compared is less than 10 units. This is to avoid having very large percentage values for small actual differences in excess. All percentages are computed as the ratio of the difference to the largest

absolute attribute value. To these scaled values we apply performance weights and sum to attain a global performance value. A value of zero would indicate comparable performance. The more positive the number the better the modified Silver model performed. Similarly, the more negative the better the UICP model performed. Thus, for a given set of management weight factors, comparative model performance can be expressed in a single value.

Using this scaling technique, Table 5.19 provides the global performance measure values for four different management weighting systems for each of the first twenty-two stationary mean demand scenarios. The columns correspond to the management weighting system identified below the table. The corresponding attributes are ACWT, total cost, and ending excess, in that order. For example, Column 1 provides equal weighting to all three attributes, while Column 2 provides twice as much weight on ACWT as it does the other two attributes. Similarly, Column 3 provides twice as much weight on total cost and Column 4 provides twice as much weight on ending excess. Although the original comparison data for the experiments is presented in Tables 5.3 and 5.4, this type of display may be more beneficial for a decision maker. For example, if the current management philosophy considers total cost to be twice as important as either ACWT or the amount of ending excess, and that global performance measure values of 1.0 or greater provide significant incentive to warrant replacement of the current model, a decision might be made based the data in Column 3 to implement the modified Silver model even under a stationary mean demand assumption for low dollar value items with high demand. Similarly, a

decision might also be made to implement the modified Silver model under a stationary mean demand assumption for low dollar items with low or moderate demand and high variability.

Applying the same technique to non-stationary demand profiles may also be instructive even though the individual differences are more distinguishable. Table 5.20 provides global performance measure values for declining demand scenarios. The original data was presented in Tables 5.11 and 5.12. Although it is clear that the modified Silver model performs significantly better than the UICP model in each scenario, the magnitude of the improvement is more evident. The reader may also find it instructive to note that in developing Table 5.20, the mean scaled values for ACWT, total cost, and ending excess are 0.03, 2.53, and 5.78, respectively. Reading these values back through the scale provides a convenient measure of the average performance of the two models for each measure. Thus, for declining demand, one should expect a 5-10%decrease in total cost and a 40-50% reduction in excess, with no significant change in ACWT.

Tables 5.21 and 5.22 provide composite performance measures for cyclic and increasing demand scenarios, respectively. The original comparison data for these scenarios was presented in Tables 5.8 and 5.16. Again, using the mean scaled values, for cyclic demand we attain average reductions from the modified Silver model of 30 days in ACWT, approximately 20% in total cost, and about 45% in total excess. For increasing demand the mean scaled values indicate an average reduction of about 40 days in ACWT, 20% in total cost, and 5%in ending excess. Note in the latter case

ending excess is reflective of the near equivalent investment levels of the two models after returning to a stationary mean condition following the increasing trend period.

The tables provided in this section are only offered as a possible decision tool. The weighting systems used may or may not reflect current reality. In practice, a management organization may need to include other cost or criteria into the decision making process. Interpretation of the table values also requires some judgement. Certainly, a composite or global performance measure of 5.0 should be considered highly significant when read back through the scaling table. On the other hand, one may be hesitant to consider a value of less than 1.0 as significant enough to warrant any immediate changes to the system given other potential cost areas not specifically addressed. The tables also provide important comparative information for decision making. For example, based on Column 2 for steep decline in Table 5.20, it is evident that if customer service (ACWT) is management's primary concern then perhaps there are only marginal gains to be had by implementing the new model. On the other hand, based on Column 4 of the same table, if management's primary focus is on reducing excess following a period of decline, there is much more to be gained by implementing the new model.

TABLE 5.19. COMPOSITE PERFORMANCE MEASURE BASED ON MANAGEMENT WEIGHTING CRITERIA FOR STATIONARY DEMAND.

Exp #	Mean	Variance	Low Dollar Value				High Dollar Value			
			1	2	3	4	1	2	3	4
1, 12	0.25 (P)	0.25	.67	.80	.80	.40	.67	.80	.80	.40
2, 13	1 (P)	1	-.33	-.60	-.20	-.20	0.00	0.00	0.00	0.00
3, 14	4 (N)	Low (2.6)	.33	.20	.20	.60	.67	1.20	.40	.40
4, 15	4 (N)	Mod (10.2)	0.00	0.00	0.00	0.00	.33	.20	.20	.60
5, 16	4 (N)	High (31.4)	1.00	.60	1.0	1.4	0.00	-.40	0.0	.40
6, 17	12 (N)	Low (23)	1.00	0.60	0.60	1.80	1.00	.60	.60	1.80
7, 18	12 (N)	Mod (92)	.67	.80	.80	.40	1.33	1.2	.80	2.00
8, 19	12 (N)	High (282)	1.00	1.00	1.00	1.00	0.33	.20	.20	.60
9, 20	25 (N)	Low (100)	1.00	.60	1.00	1.40	1.67	1.00	1.00	3.00
10, 21	25 (N)	Mod (400)	2.00	1.60	2.00	2.40	.67	.80	.40	.80
11, 22	25 (N)	High (1225)	1.33	1.20	1.60	1.20	0.00	0.00	0.00	0.00

(P) - Poisson (N) - Normal ACWT / Total Cost / Ending Excess Weighting:
Col.1=0.33/0.33/0.33 Col. 2=0.50/0.25/0.25 Col. 3=0.25/0.50/0.25 Col. 4=0.25/0.25/0.50

TABLE 5.20. COMPOSITE PERFORMANCE MEASURE BASED ON MANAGEMENT WEIGHTING CRITERIA FOR DECLINING DEMAND.

Exp #	Mean	Variance	Steep Decline				Slow Decline			
			1	2	3	4	1	2	3	4
45,41	0.25 (P)	0.25	2.33	1.40	2.20	3.40	2.33	1.40	2.20	3.40
53,49	0.25 (P)	0.25	1.67	1.00	1.00	3.00	2.33	1.40	2.20	3.40
61,57	0.25 (P)	0.25	2.33	1.40	2.20	3.40	2.33	1.40	2.20	3.4
69,73	0.25 (P)	0.25	2.67	1.60	2.80	3.60	2.00	1.20	2.00	2.80
46,42	25 (N)	Low (100)	3.33	2.00	3.20	4.80	3.33	2.00	3.20	4.80
54,50	25 (N)	Low (100)	3.33	2.00	3.20	4.80	3.33	2.00	3.20	4.80
62,58	25 (N)	Low (100)	3.00	1.80	2.60	4.60	3.33	2.00	3.20	4.80
70,66	25 (N)	Low (100)	3.67	2.20	3.40	5.40	3.67	2.20	3.80	5.00
47,43	25 (N)	Mod (400)	3.00	1.80	3.00	4.20	3.00	1.80	3.00	4.20
55,51	25 (N)	Mod (400)	2.67	1.60	2.40	4.00	3.00	1.80	3.00	4.20
63,59	25 (N)	Mod (400)	2.67	1.60	2.40	4.00	3.00	1.80	3.00	4.20
71,67	25 (N)	Mod (400)	3.33	2.40	3.20	4.40	3.00	1.80	3.00	4.20
48,44	25 (N)	High (1225)	2.67	1.60	2.80	3.60	2.67	1.60	2.80	3.60
56,52	25 (N)	High (1225)	2.33	1.40	2.20	3.40	2.33	1.40	2.20	3.40
64,60	25 (N)	High (1225)	2.33	1.40	2.20	3.40	2.33	1.40	2.20	3.40
72,68	25 (N)	High (1225)	3.00	1.80	3.00	4.20	2.67	1.60	2.80	3.60

(P) - Poisson (N) - Normal ACWT / Total Cost / Ending Excess Weighting:
Col.1=0.33/0.33/0.33 Col. 2=0.50/0.25/0.25 Col. 3=0.25/0.50/0.25 Col. 4=0.25/0.25/0.50

TABLE 5.21. COMPOSITE PERFORMANCE MEASURE BASED ON MANAGEMENT WEIGHTING CRITERIA FOR CYCLIC DEMAND.

Exp #	Mean	Variance	Steep Decline				Slow Decline			
			1	2	3	4	1	2	3	4
33,37	0.25 (P)	0.25	3.00	3.00	2.20	3.80	4.00	4.00	3.20	4.80
34,38	4 (N)	Low (2.6)	4.67	4.40	4.40	5.20	6.33	5.80	6.60	6.60
35.39	4 (N)	Mod (10.2)	4.33	4.20	4.20	4.60	5.33	5.60	4.80	5.60
36,40	4 (N)	High (31.4)	5.33	5.20	5.60	5.20	5.00	6.00	6.40	5.60

(P) - Poisson (N) - Normal ACWT / Total Cost / Ending Excess Weighting:
Col.1=0.33/0.33/0.33 Col. 2=0.50/0.25/0.25 Col. 3=0.25/0.50/0.25 Col. 4=0.25/0.25/0.50

TABLE 5.22. COMPOSITE PERFORMANCE MEASURE BASED ON MANAGEMENT WEIGHTING CRITERIA FOR INCREASING DEMAND.

Exp #	Mean	Variance	Steep Decline				Slow Decline			
			1	2	3	4	1	2	3	4
81,77	0.25 (P)	0.25	1.67	3.00	1.00	1.00	0.67	1.20	0.40	0.40
89,85	0.25 (P)	0.25	1.67	3.00	1.00	1.00	1.33	2.40	0.80	0.80
82,78	4 (N)	Low (2.6)	5.67	5.80	6.20	5.00	3.67	3.80	4.20	3.00
90,86	4 (N)	Low (2.6)	5.33	5.60	6.00	4.40	4.00	4.00	4.40	3.60
83,79	4 (N)	Mod (10.2)	5.00	5.80	5.80	3.40	3.30	3.60	4.0	2.40
91,87	4 (N)	Mod (10.2)	5.00	5.80	5.80	3.40	4.00	4.40	4.80	2.80
84,80	4 (N)	High (31.4)	5.33	6.40	6.0	3.60	3.30	4.0	4.0	2.0
92,88	4 (N)	High (31.4)	5.33	6.40	6.00	3.60	3.67	4.20	4.60	2.20

(P) - Poisson (N) - Normal ACWT / Total Cost / Ending Excess Weighting:
Col.1=0.33/0.33/0.33 Col. 2=0.50/0.25/0.25 Col. 3=0.25/0.50/0.25 Col. 4=0.25/0.25/0.50

VI. SUMMARY, CONCLUSIONS, AND RECOMMENDATIONS

A. SUMMARY

The Navy's existing UICP (s,S) inventory model for demand based consumable items is based in part on the economic order quantity. One key assumption of the model is that mean demand remains stationary over time. In reality, this assumption is often violated. In order to partially compensate for this, the reorder point and order up-to-level are periodically recalculated using newly forecasted mean demand. This procedure works well enough as long as the increase or decrease in mean demand is gradual. However, if the trend is steep, the existing model can significantly under- or overestimate s and S. In an increasing demand environment, the result can be exceptionally poor levels of customer service. In a declining demand environment, such as that associated with a major ship decommissioning program, the result can be the creation of substantial amounts of excess inventory.

In this thesis, we have proposed an alternative inventory model which does not rely on the assumption of a stationary mean. The model is an extension of Silver's lot sizing heuristic for stochastic demand with a time varying mean [Ref. 7]. Our modified Silver model includes provisions for stochastic lead times and a modified version of the Silver-Meal heuristic for determining the length of an order cycle. The model uses the existing UICP forecasting system to obtain a single period forecast but allows the inclusion of predicted or known increases or declines in future forecasts.

Decision rules for implementation, including recommendations for the values of various variables and parameters, have also been discussed.

Evaluation of the modified Silver model is based on a Monte Carlo simulation. The baseline measurement is the performance of the current Navy Uniform Inventory Control Program (UICP) model for consumable items under the same simulated demand scenarios. Both simulations approximate the inventory management of a single item for as many as 120 quarters. The simulation experiments include a variety of run characteristics, system parameter settings, and generated demand profiles. Testing of the modified Silver model using a stationary mean demand forecast demonstrated comparable or slightly improved performance over the UICP model. This supports the assertion that the models are nearly equivalent under the assumption that mean demand is stationary.

B. CONCLUSIONS

Simulation tests of each model clearly demonstrated that the modified Silver model outperforms the existing UICP model when mean demand is varying and estimates of the varying mean are included in the forecasts. In declining demand scenarios, the modified Silver model significantly reduced both excess inventory and total cost with no reduction in average customer wait time. In increasing demand scenarios, the model significantly reduced average customer wait time at an overall lower total cost.

In many cases inventory managers are presented information regarding program changes which will have a significant impact on the future mean rate of demand. Being able to use this information in determining inventory levels is critical if the manager wishes to maintain an adequate level of customer service while avoiding embarrassing and costly excess or deficit inventory positions. The current UICP inventory model lacks the capability to accept a varying mean demand forecast. The modified Silver model is a suitable alternative which is both simple to understand and readily implementable within the existing inventory information system.

C. RECOMMENDATIONS

This study has not examined the issue of varying the point in time at which modified forecasting is implemented when using the modified Silver model. Under normal circumstances, knowledge of future trends may not be available until nearly the moment the trend commences. Therefore, it is recommended that additional research be done to determine the extent to which the performance of the modified Silver model may be degraded by late forecasts of changes in mean demand. It should be noted though, since the UICP and modified Silver models are generally comparable when the same steady state forecasting process is used, one can reasonably expect the modified Silver model with late forecasting to perform no worse than the current model.

There is strong evidence to suggest that the use of forecasted mean absolute deviation to estimate the variance of forecasted lead time demand is erroneous in the

case of very low demand items. Additional research should include the formulation of a power rule for this variance similar to that used in UICP.

The current model has been developed and tested only for consumable items. Since the current UICP model includes an economic reorder quantity determination for the procurement of repairable items, some future research should examine the potential applicability and integration of the modified Silver model into the UICP repairables model.

In addition to the modified Silver model described in this thesis, other deterministic inventory models exist that may, with appropriate modifications, perform well for time varying stochastic demand with stochastic lead times. One such model, the Wagner-Whitin algorithm, is known to guarantee optimality for the deterministic case in terms of minimizing the total cost of ordering and holding inventory [Ref. 9: p. 227], although it can be expected to require significantly increased computation time and has more theoretical complexity. A list and description of such alternative models is found in Tersine [Ref. 35: Chapter 4], or Silver and Peterson [Ref. 9: Chapter 6].

APPENDIX A. SIMULATION CODE

```
{$M $4000,0,0} {$r+} {$N+,E+} {$G+} {Q+}
program Mod_Silver_Simulator (input,output);

uses dos, crt, toolbox, unirand, pqueue;

type quarterArray=array [1..120] of real;
    weeklyArray=array [1..1560] of real;
    qtrIntArray=array [1..120] of integer;
    changeRealArry = array [1..10] of real;
    changeIntArry = array [1..10] of integer;
    descriptType=string[40];
    statRecord = record
                 Mean:real;
                 Variance:real;
                 CIHigh:real;
                 CILow:real;
                end;
    qtrStatArry=array [1..120] of statRecord;

const COEFF1=1.386;
     POWER1=0.746;
     COEFF2=3.869;
     POWER2=1.378;
     MAXPLT=14.0;
     MINPLT=2.0;
     ERROR=1.00000000000000E-0010;

var wklyObserv:weeklyArray;
   observ, frcst, mad, meanDmdArry, varDmdArry:quarterArray;
   stepIndArry, trndIndArry,mkCodeArry:qtrIntArray;
   qtrACWTBOArry,qtrACWTArry,qtrSMAArry,qtrInvestArry,qtrInappArry:qtrStatArry;
   cumACWTBOArry,cumACWTArry,cumSMAArry:qtrStatArry;
   observType,distrType,outputType,seedType,wkDataType,qtrDataType,
   repStatType,frcstDataType,silverSSType:char;
   numberRep,i,n,s,numberOfReps,numberOfQtrs,numberOfWks,markCode,initInv,
   initOS,initOrders,simCount,startSSQtr,endSSQtr,initSSOH,initSSOS,
   initSSOrders:integer;
   meanDemand, varDemand:real;
   trendOn,StepOn,nmbrSteps, nmbrTrends,ROLowConst,maxQtrs,minQtrs,maxDecl,
   seedIndex,negBinS:integer;
   TWUS:longint;
   unitPrice,PLT,P1,adminCost,obsol,timePref,storage,shortCost,
   frcstErrCoeff,bufferMult,PLTSigMuRatio:real;
```

```
    inputfile,outputfile:text;
    realval,negBinP:real;
    stop:boolean;
    startstep, startrnd, endtrnd: changeIntArry;
    stepmult, trendcoeff, trendpower: changeRealArry;
    hour1,minute1,second1,hdSec1,hour2,minute2,second2,hdSec2:word;
    outFileName:string;
    OSHeap,BOHeap:PriorityQueueType;
    ACWTBO,ACWT,SMA,Invest,orderCount,lastOH,lastOS,totalCost,
    inappAsset,inappVal:real;
    simACWTBO, simACWT, simSMA, simInvest, simOrderCount,simLastOH,
    simLastOS,simACWTBOVar,simACWTVar,simSMAVar,simInvestVar,simOrderCountVar,
    simLastOHVar,simLastOSVar,simTotalCost,simTotalCostVar,simInapp,simInappvar,
    simInappVal,simInappValVar,simInitSSOH,simInitSSOS,simInitSSOrders,
    simInitSSOHVar,simInitSSOSVar,simInitSSOrdersVar:real;
    runDescript:descriptType;
    currSeed:longint;

procedure Frontscreen;

begin
  clrscr;
  writeln;
  writeln;
  writeln;
  writeln;
  writeln;
  writeln;
  writeln ('                ************************************');
  writeln ('                *          MODIFIED                *');
  writeln ('                *        SILVER MODEL              *');
  writeln ('                *          SIMULATOR               *');
  writeln ('                *        FOR CONSUMABLES           *');
  writeln ('                *                                  *');
  writeln ('                *    G. C. Robillard LT,SC         *');
  writeln ('                *                                  *');
  writeln ('                *       Revised: 9/01/93           *');
  writeln ('                ************************************');
  Delay(1500);  {For 1500 ms}
  clrscr;
end;

procedure runtype (var distrType,outputType,wkDataType,qtrDataType,
                   frcstDataType,repStatType,silverSSType:char;
                 var numberOfQtrs,numberOfWks,numberOfReps,negBinS,
                   seedIndex,startSSQtr,endSSQtr:integer;
                 var meanDemand, varDemand,negBinP:real;
                 var inputfile,outputfile: text;
                 var frcst,mad: quarterArray;
                 var seeds:seedArryType;
```

```
                var outFileName:string;
                var runDescript:descriptType);
var done: boolean;
   i,maxStart:integer;
   demandInFile: string;

begin
  writeln;
  writeln ('  *** THIS SCREEN WILL ALLOW SELECTION OF RUN TYPE OPTIONS ***');
  done:=FALSE;
  writeln;
  writeln; writeln;
  write  ('Enter the number of replications (from 1 to 20000) to be run : ' );
  numberOfReps:=Get_Integer(1,20000);
  writeln;
  repeat
    writeln ('Random Number Generator Seed Selection: ');
    writeln;
    writeln ('  1 - Default Seeds (unique seed for each replication)');
    writeln ('  2 - Select Seeds (max number of replications is 100)');
    writeln;
    write ('Choice: ');
    seedtype:=readkey;
    writeln (seedtype);
    writeln;
    case seedtype of
      '1': begin
           done:=TRUE;
           maxStart:=20001-NumberOfReps;
           write('Enter Random Seed Start Index (1 to ',maxStart:2,'): ');
           seedIndex:=Get_Integer(1,maxStart);
           end;
      '2': begin
           done:=TRUE;
           if NumberOfReps > 100 then NumberOfReps:=100;
           for i := 1 to numberOfReps do begin
             write ('Enter Seed value for replication ',i,' : ');
             seeds[i]:=Get_LongInt(1,2147483646);
             writeln;
          end;  {for}
          end
    end
  until done=TRUE;
  clrscr;
  writeln ('        **** RUN SELECTION OPTIONS CONTINUED ****');
  writeln;
  writeln;
  write('Enter Run Description: ');
  readln (runDescript);
  writeln;
```

```
  write  ('Enter the number of simulation quarters (max 120): ' );
  numberOfQtrs:=Get_Integer(1,120);
  numberOfWks:=13*NumberOfQtrs;
  writeln;
  write  ('Enter the start of simulation SS (collect stats) quarter (max ',numberOfQtrs:3,'): '
);
  startSSQtr:=Get_Integer(1,numberOfQtrs);
  writeln;
  write  ('Enter the end of simulation SS (collect stats) quarter (max ',numberOfQtrs:3,'): '
);
  endSSQtr:=Get_Integer(startSSQtr,numberOfQtrs);
  writeln;
  done:=FALSE;
  repeat
    writeln ('Type of Distribution: ');
    writeln;
    writeln ('  1 - Normal');
    writeln ('  2 - Poisson');
    writeln ('  3 - Neg Binomial');
    writeln;
    write ('Choice: ');
    distrType:=readkey;
    writeln (distrType);
    writeln;
    case distrType of
      '1': begin
          done:=TRUE;
          write ('Enter quarterly mean demand: ');
          meanDemand:=Get_Real(0.0001,999999.0);
          writeln;
          write ('Enter demand variance: ');
          varDemand:=Get_Real(0.0001,999999.0);
          writeln
          end;
      '2': begin
          done:=TRUE;
          write ('Enter quarterly mean demand: ');
          meanDemand:=Get_Real(0.0001,999999.0);
          varDemand:=meanDemand;
          writeln;
          end;
      '3': begin
          done:=TRUE;
          repeat
            writeln;
            write ('Enter parameter p  (0 < p < 1): ');
            negBinP:=Get_Real(0.0001,0.9999);
            writeln;
            write ('Enter parameter s (s =  1,2,3 ... ) : ');
            negBinS:=Get_Integer(1,100);
```

```
            writeln;
            meanDemand:=(negbinS+(1-negBinP))/negBinP;
            varDemand:=(negBinS+(1-negBinP))/(sqr(negBinP));
            writeln('The quarterly mean is:     ',meanDemand:8:2);
            writeln('The demand variance is: ',varDemand:8:2);
            writeln;
            write('Change Initial Neg Binomial Parameters? (Y or N): ');
          until not(Get_Answer);
          end;
    end
until done=TRUE;
frcst[1]:=meanDemand;
mad[1]:=COEFF1*exp(POWER1*ln(frcst[1]));
done:=FALSE;
clrscr;
writeln ('          **** RUN SELECTION OPTIONS CONTINUED ****');
repeat
    writeln;
    writeln ('Send Output to: ');
    writeln;
    writeln ('  1 - Screen');
    writeln ('  2 - File');
    writeln;
    write ('Choice: ');
    outputType:=readkey;
    writeln (outputType);
    case outputType of
      '1': begin
             done:=TRUE;
             assign(outputfile,'con');
           end;
      '2': begin
             done:=TRUE;
             repeat
               writeln;
               write ('Enter Path and Filename: ');
               readln (outFileName);
               writeln;
               writeln ('Path and FileName entered: ',outFileName);
               writeln;
               write ('Change Path and FileName entered? (Y or N): ');
             until not(Get_Answer);
             assign(outputfile,outFileName);
           end;
      end;
until done=TRUE;
clrscr;
writeln ('          **** RUN SELECTION OPTIONS CONTINUED ****');
wkDataType:='0';
writeln;
```

```
  write('Include Weekly SDR Data? (Y or N): ');
  if Get_Answer then  wkDataType:='1';
  qtrDataType:='0';
  writeln;
  write('Include Quarterly SDR Data? (Y or N): ');
  if Get_Answer then  qtrDataType:='1';
  frcstDataType:='0';
  writeln;
  write('Include Quarterly demand and forecast Data? (Y or N): ');
  if Get_Answer then  frcstDataType:='1';
  repStatType:='0';
  writeln;
  write('Include Replication Statistics? (Y or N): ');
  if Get_Answer then  repStatType:='1';
  silverSSType:='0';
  writeln;
  write('Run Silver Model Using Fixed Future Forecasts? (Y or N): ');
  if Get_Answer then  silverSSType:='1';
end;


procedure RunAgain (var outputfile:text;var runDescript:descriptType;
                    outputType:char;
                    var frcst,mad:quarterArray;
                    var stop:boolean;
                    var outFileName:string);

var demandInFile: string;
   done1:boolean;
begin
  stop:=FALSE;
  clrscr;
  writeln ('        **** RE-RUN SIMULATION OPTIONS SCREEN ****');
  writeln;
  writeln('Re-running the simulation will maintain the same run-type parameters, but will');
  writeln('allow the user to change the destination (output) file and vary NIIN');
  writeln('and model parameters.');
  writeln;
  write('Do you wish to re-run the simulation? (Y or N): ');
  if Get_Answer then begin
    writeln;
    write('Change Run Description? (Y or N): ');
      if Get_Answer then begin
         writeln;
         write ('Enter Run Description: ');
         readln (runDescript);
      end;
    if outputType='2' then begin
      writeln;
      write('Change Output File? (Y or N): ');
```

```
        if Get_Answer then begin
          repeat
            writeln;
            write ('Enter Output Path and Filename: ');
            readln (outFileName);
            writeln;
            writeln ('Path and FileName entered: ',outFileName);
            writeln;
            write ('Change Path and FileName entered? (Y or N): ');
          until not(Get_Answer);
          assign(outputfile,outFileName);
        end;
      end;
    end else begin
         stop:=TRUE;
    end;
    clrscr;
end;


procedure InitInput(var unitPrice,PLT,P1,adminCost,obsol,timePref,storage,shortCost,
                frcstErrCoeff,bufferMult,PLTSigMuRatio,meanDemand:real;
                var ROLowConst,maxQtrs,minQtrs,maxDecl,
                   initInv,initOS,initOrders:integer);

begin
  unitPrice:=100.00;    {unit price}
  PLT:= 8;              {procurement leadtime}
  P1:=0.1;              {probability of stockout}
  adminCost:=850.00;
  obsol:=0.12;
  timePref:=0.10;
  storage:=0.01;
  shortCost:=1000.00;
  ROLowConst:=1;
  maxQtrs:=6;
  minQtrs:=1;
  maxDecl:=4;
  frcstErrCoeff:=0.0;
  bufferMult:=0.5;
  PLTSigMuRatio:=(sqrt(1.57*PLT))/PLT;
  initInv:=round(meanDemand * (PLT+3));
  if initInv < 1 then initInv:=1;
  initOS:=round(3 * meanDemand);
  if initOS < 1 then initOS:=1;
  initOrders:=1;
end;

procedure InputEdit(var unitPrice,PLT,P1,adminCost,obsol,timePref,storage,shortCost,
                frcstErrCoeff,bufferMult,PLTSigMuRatio,meanDemand:real;
```

```
            var ROLowConst,maxQtrs,minQtrs,maxDecl,initInv,initOS,initOrders:integer);

var editChoice:char;
   done:boolean;
   frcstErrString:string;
begin
  done:=FALSE;
  repeat
  clrscr;
  writeln('  **** THIS SCREEN ALLOWS EDITING OF DEFAULT NIIN INPUT PARAMETERS
****');
  writeln;
  writeln;
  writeln (' A.  Unit Price     : ',unitPrice:8:2,'   J.  Admin Order   : ',adminCost:8:2);
  writeln (' B.  Buffer Mult (B): ',bufferMult:8:2,'   K.  R/O Constr    : ',ROLowConst:8);
  writeln (' C.  Frcst Error (C): ',frcstErrCoeff:8:2,'   L.  Obsol Rate    : ',obsol:8:2);
  writeln (' D.  PLT Sig/Mu     : ',PLTSigMuRatio:8:2,'   M.  Time Pref Rate: ',timePref:8:2);
  writeln (' E.  Max Qtrs       : ',maxQtrs:8,'   N.  Storage Rate  : ',storage:8:2);
  writeln (' F.  Max Decl Qtrs  : ',maxDecl:8,'   O.  Shortage Cost : ',shortCost:8:2);
  writeln (' G.  Min Qtrs       : ',minQtrs:8,'   P.  Init Inv OH   : ',initInv:8);
  writeln (' H.  Procur LT      : ',PLT:8:2,'   Q.  Init Qty OS   : ',initOS:8);
  writeln (' I.  Risk           : ',P1:8:2,'   R.  Init Num Order: ',initOrders:8);
  writeln;
  writeln ('     Hit ENTER to accept current values ');
  write ('       or letter of field to change: ');
  editChoice:=upcase(readkey);
  writeln(editChoice);
  case editChoice of
     'A' : begin
           writeln;
           write ('Enter new Unit Price: ');
           unitPrice:=Get_Real(0.0,999999.0);
         end;
     'B' : begin
           writeln;
           write ('Enter new Buffer Multiple: ');
           bufferMult:=Get_Real(0.0,999999.0);
         end;
     'C' : begin
           writeln;
           writeln('*** Note: Default = 0.0 (Calculate Using MAD) ***');
           writeln;
           write ('Enter new Forecast Error Coeff of Variation: ');
           frcstErrCoeff:=Get_Real(0.0,999999.0);
         end;
     'D' : begin
           writeln;
           write ('Enter new PLT Std Deviation to Mean Ratio: ');
           PLTSigMuRatio:=Get_Real(0.0,3.0);
         end;
```

```
'E' : begin
        writeln;
        write ('Enter new Maximum Quarters Constraint: ');
        maxQtrs:=Get_Integer(1,16);
      end;
'F' : begin
        writeln;
        write ('Enter new Maximum Quarters during Decline Constraint: ');
        maxDecl:=Get_Integer(1,maxQtrs);
      end;
'G' : begin
        writeln;
        write ('Enter new Minimum Quarters Ordering Constraint: ');
        minQtrs:=Get_Integer(1,maxQtrs);
      end;
'H' : begin
        writeln;
        write ('Enter new Procurement Leadtime Forecast: ');
        PLT:=Get_Real(0.0,15.0);
        PLTSigMuRatio:=(sqrt(1.57*PLT))/PLT;
        initInv:=round(meanDemand*PLT);
        if initInv < 1 then initInv:=1;
      end;
'I' : begin
        writeln;
        write ('Enter new Probability of Stockout (Risk): ');
        P1:=Get_Real(0.0,0.9999);
      end;
'J' : begin
        writeln;
        write ('Enter new Admin Order Cost: ');
        adminCost:=Get_Real(0.0,999999.0);
      end;
'K' : begin
        writeln;
        write ('Enter new System Reorder Level Low Limit Constraint: ');
        ROLowConst:=Get_Integer(0,9999);
      end;
'L' : begin
        writeln;
        write ('Enter new Obsolescence Rate: ');
        obsol:=Get_Real(0.0,999999.0);writeln;
      end;
'M' : begin
        writeln;
        write ('Enter new Time Preference Rate: ');
        timePref:=Get_Real(0.0,99999.0);
      end;
'N' : begin
        writeln;
```

```
            write ('Enter new Storage Cost Rate: ');
            storage:=Get_Real(0.0,99999.0);
          end;
      'O' : begin
            writeln;
            write ('Enter new Shortage Cost: ');
            shortCost:=Get_Real(0.0,99999.0);
          end;
      'P' : begin
            writeln;
            write ('Enter new Initial Inventory On Hand Qty: ');
            initInv:=Get_Integer(0,9999);
          end;
      'Q' : begin
            writeln;
            write ('Enter new Initial OutStanding Qty: ');
            initOS:=Get_Integer(0,9999);
          end;
      'R' : begin
            writeln;
            write ('Enter new Initial Number of Orders: ');
            initOrders:=Get_Integer(0,9999);
          end;

      chr(13): done:=TRUE
  end;
  until done=TRUE;
  clrscr;
end;


function GetMarkCode (t,oldMark:integer; frcst, unitPrice:real):integer;

begin
  if t=1 then begin
    if frcst < 0.25 then getMarkCode:=0;
    if (frcst >= 0.25) and (frcst < 2.0) then begin
      if (unitPrice >= 300.00) then begin
        getMarkCode:=3;
      end else begin
        getMarkCode:=1;
      end;
    end;
    if frcst >= 2.0 then begin
      if (unitPrice*frcst) >= 600.0 then begin
        getMarkCode:=4;
      end else begin
        getMarkCode:=2
      end;
    end;
```

```
  end else begin
    getMarkCode:=oldMark;
    if oldMark = 0 then begin
      if frcst >= 0.5 then begin
        if (unitPrice >= 300.00) then begin
          getMarkCode:=3;
        end else begin
          getMarkCode:=1;
        end;
      end;
      if frcst >=3 then begin
        if (unitPrice*frcst) >= 600.0 then begin
          getMarkCode:=4;
        end else begin
          getMarkCode:=2
        end;
      end;
    end;
    if (oldMark=1) or (oldMark=3) then begin
      if frcst >=3 then begin
        if (unitPrice*frcst) >= 600.0 then begin
          getMarkCode:=4;
        end else begin
          getMarkCode:=2
        end;
      end else if unitPrice <= 200 then begin
          getMarkCode:=1;
      end else if unitPrice >= 400 then begin
          getMarkCode:=3;
      end;
      if frcst <= 0.25 then getMarkCode:=0;
    end;
    if (oldMark=2) or (oldMark=4) then begin
      if frcst <= 1.0 then begin
        if (unitPrice >= 300.00) then begin
          getMarkCode:=3;
        end else begin
          getMarkCode:=1;
        end;
      end else if (unitPrice*frcst) >= 800.00 then begin
          getMarkCode:=4;
      end else if (unitPrice*frcst) <= 400.00 then begin
         getMarkCode:=2;
      end;
      if frcst <= 0.25 then getMarkCode:=0;
    end;
  end
end;

procedure InitializeStatArrays(var qtrACWTBOArry,qtrACWTArry,qtrSMAArry,
```

```
                    qtrInvestArry,qtrInappArry,cumACWTBOArry,
                    cumACWTArry,cumSMAArry:qtrStatArry);

var t:integer;

begin
  for t:=1 to numberOfQtrs do begin
    qtrACWTBOArry[t].Mean:=0.0; qtrACWTBOArry[t].Variance:=0.0;
    qtrACWTBOArry[t].CIHigh:=0.0; qtrACWTBOArry[t].CILow:=0.0;
    qtrACWTArry[t].Mean:=0.0; qtrACWTArry[t].Variance:=0.0;
    qtrACWTArry[t].CIHigh:=0.0; qtrACWTArry[t].CILow:=0.0;
    qtrSMAArry[t].Mean:=0.0; qtrSMAArry[t].Variance:=0.0;
    qtrSMAArry[t].CIHigh:=0.0; qtrSMAArry[t].CILow:=0.0;
    qtrINVESTArry[t].Mean:=0.0; qtrINVESTArry[t].Variance:=0.0;
    qtrINVESTArry[t].CIHigh:=0.0; qtrINVESTArry[t].CILow:=0.0;
    qtrInappArry[t].Mean:=0.0; qtrInappArry[t].Variance:=0.0;
    qtrInappArry[t].CIHigh:=0.0; qtrInappArry[t].CILow:=0.0;
    cumACWTBOArry[t].Mean:=0.0; cumACWTBOArry[t].Variance:=0.0;
    cumACWTBOArry[t].CIHigh:=0.0; cumACWTBOArry[t].CILow:=0.0;
    cumACWTArry[t].Mean:=0.0; cumACWTArry[t].Variance:=0.0;
    cumACWTArry[t].CIHigh:=0.0; cumACWTArry[t].CILow:=0.0;
    cumSMAArry[t].Mean:=0.0; cumSMAArry[t].Variance:=0.0;
    cumSMAArry[t].CIHigh:=0.0; cumSMAArry[t].CILow:=0.0;
  end;
end;

procedure InitializeArrays (var observ,meanDmdArry,varDmdArry:quarterArray;
                    var  stepIndArry, trndIndArry,mkCodeArry: qtrIntArray;
                    numberOfQtrs,numberOfWks:integer;
                    meanDemand:real;
                    var wklyObserv:weeklyArray);
var t:integer;

begin
  for t:=1 to numberOfQtrs do begin
    observ[t]:=0.0;
    meanDmdArry[t]:=0.0;
    varDmdArry[t]:=0.0;
    stepIndArry[t]:=0;
    trndIndArry[t]:=0;
    mkCodeArry[t]:=0;
  end;
  for t:=1 to (numberOfWks) do begin
    wklyObserv[t]:=0.0;
  end;
end;


procedure LoadObserv (var observ,frcst,mad,meanDmdArry,varDmdArry:quarterArray;
                var wklyObserv:weeklyArray;
```

```
                observType,distrType:char;
                numberOfQtrs,numberOfWks,repNum,simCount:integer;
                var trendInd,stepInd,nmbrSteps, nmbrTrends:integer;
                meanDemand, varDemand:real;
                var inputfile:text;
                var startstep, startrnd, endtrnd: changeIntArry;
                var stepmult, trendcoeff, trendpower: changeRealArry);


var SS:char;
   i, t, min, startQtr, endQtr,observWeek,s:integer;
   randnorm, currMeanDmd, initTrendMean, coeffVar,qtrCum,qtrMean,
   wkObserv,qtrObserv,p:real;
   demandInFile:string;

begin
  if (repNum = 1) and (simCount = 1) then begin
    for i:=1 to 10 do begin
      startstep[i]:=0; startrnd[i]:=0; endtrnd[i]:=0;
      stepmult[i]:=0.0; trendcoeff[i]:=0.0; trendpower[i]:=0.0;
    end;
    nmbrSteps:=0;
    nmbrTrends:=0;
  end; {if}
  currMeanDmd:=meanDemand;
  coeffVar:=sqrt(varDemand)/meanDemand;
  for t:=0 to (numberOfQtrs) do begin
    if (t=0) and (repNum = 1) and (simCount=1) then begin
      SS:='Y';
      writeln;
      write('Do you wish to vary mean demand rate over time? (Y or N): ');
      if Get_Answer then begin
        SS:='N';
        stepInd:=0;
        trendInd:=0;
        clrscr;
        writeln;
        writeln ('              *** Mean Demand Variants ***   ');
        writeln;
        writeln ('You have the option to vary mean demand rate over time.  If the normal');
        writeln ('distribution was selected, variance will also change to maintain your');
        writeln ('original variance to mean ratio.  You may choose between step change');
        writeln ('or trend or any combination of the events.  If more than one event is');
        writeln ('chosen to occur at the same time, step changes will occur first.');
        writeln ('A maximum of 10 occurances of each event is allowed.  Time of');
        writeln ('variation is specified by quarter.');
        writeln ;

        SS:='Y';
        write ('Do you still wish to vary mean demand rate over time? (Y or N): ');
```

```
if Get_Answer then begin
  SS:='N';
  clrscr;
  writeln('              *** Step Changes Screen ***');
  writeln;
  write ('Do you wish to have step increases or decreases? (Y or N): ');
  if Get_Answer then stepInd:=1;
  if stepInd=1 then begin
    writeln;
    write('Enter the number of steps changes desired (max 10): ');
    nmbrSteps:=Get_Integer(1,10);
    writeln;
    writeln('The step function is of the form:  Mean(t) = A * Mean(t-1).');
    writeln('You must specify the value of "A" for each step.');
    min:=1;
    for i:=1 to nmbrSteps do begin
      writeln;
      writeln ('Step ',i,':');
      writeln;
      write ('Step Qtr: ');
      startQtr:=Get_Integer(min,numberOfQtrs);
      startstep[i]:=startQtr;
      writeln;
      write ('Step Multiplier (A): ');
      stepmult[i]:=Get_Real(0.00001,9999.0);
      writeln;
      min:=startQtr;
    end;
  end;
  clrscr;
  writeln('              *** Trend Setting Screen ***');
  writeln;
  write ('Do you wish to have trends? (Y or N):');
  if Get_Answer then trendInd:=1;
  if trendInd=1 then begin
    writeln;
    write('Enter the number of trend periods desired (max 10): ');
    nmbrtrends:=Get_Integer(1,10);
    writeln;
    writeln('The trend function is of the form:');
    writeln('              Mean(t) = InitTrendMean * ( 1 + A * t(0) ** B)');
    writeln('where t(0) is reset to "1" at the beginning of each trend period');
    writeln('and InitTrendMean is the Mean at the beginning of the trend period.');
    writeln('Parameters A and B must be specified for each trend period.');
    min:=1;
    for i:=1 to nmbrtrends do begin
      writeln;
      writeln ('Trend ',i,':');
      writeln;
      write ('Start Qtr: ');
```

```
            startQtr:=Get_Integer(min,numberOfQtrs);
            startrnd[i]:=startQtr;
            writeln;
            write ('End Qtr: ');
            endQtr:=Get_Integer(startQtr,numberOfQtrs);
            endtrnd[i]:=endQtr;
            writeln;
            write ('Trend coefficent (A): ');
            trendcoeff[i]:=Get_Real(-9999.0,9999.0);
            writeln;
            write ('Trend power (B): ');
            trendpower[i]:=Get_Real(-9999.0,9999.0);
            writeln;
            min:=endQtr+1;
          end;
        end;
      end;
    clrscr;
    end;
  end else if t > 0 then begin
    if SS='Y' then begin
      meanDmdArry[t]:=meanDemand;
      if (distrType='1') or (distrType='3') then begin
        varDmdArry[t]:=varDemand;
      end else begin
        varDmdArry[t]:=currMeanDmd;
      end;
    end else begin
      if stepInd = 1 then begin
        for i:=1 to nmbrSteps do begin
          if t = startstep[i] then currMeanDmd:=stepmult[i]*currMeanDmd;
        end;
      end;
      if trendInd = 1 then begin
        for i:=1 to nmbrTrends do begin
          if t = startrnd[i] then initTrendMean:=currMeanDmd;
          if (t >= startrnd[i]) and (t <= endtrnd[i]) then begin
            currMeanDmd:=initTrendMean*(1+trendcoeff[i]*
                    (exp(trendpower[i]*ln(t-startrnd[i]+1))));
            if currMeanDmd < 0.0 then currMeanDmd:=0.0;
          end;
        end;
      end;
      meanDmdArry[t]:=currMeanDmd;
      if (distrType='1') or (distrType='3') then begin
        varDmdArry[t]:=sqr(coeffVar*currMeanDmd);
      end else begin
        varDmdArry[t]:=currMeanDmd;
      end;
    end;
```

```
        if distrType = '1' then begin
          randnorm: = GetNormal;
          qtrObserv: = round(meanDmdArry[t] + (randnorm * sqrt(varDmdArry[t])));
          if qtrObserv < 0.0 then qtrObserv: = 0.0;
          for i: = 1 to round(qtrObserv) do begin
            observWeek: = GetUniformInt(13);
            wklyObserv[(t-1) * 13 + observWeek]: =
                  wklyObserv[(t-1) * 13 + observWeek] + 1;
          end;
        end else if distrType = '2' then begin
           qtrObserv: = GetPoisson(meanDmdArry[t]);
           for i: = 1 to round(qtrObserv) do begin
            observWeek: = GetUniformInt(13);
            wklyObserv[(t-1) * 13 + observWeek]: =
                  wklyObserv[(t-1) * 13 + observWeek] + 1;
          end;
        end else if distrType = '3' then begin
           p: = (meanDmdArry[t])/(varDmdArry[t]);
           s: = round((sqr(meanDmdarry[t]))/(varDmdArry[t]-meanDmdArry[t]));
           if (p > ERROR) and (p < (1-ERROR)) then begin
              qtrObserv: = GetNegBin(p,s);
           end else begin
              qtrObserv: = 0.0;
           end;
           for i: = 1 to round(qtrObserv) do begin
            observWeek: = GetUniformInt(13);
            wklyObserv[(t-1) * 13 + observWeek]: =
                  wklyObserv[(t-1) * 13 + observWeek] + 1;
          end;
        end;
        observ[t]: = qtrObserv;
      end; {else,if}
    end; {for}
    clrscr;
end;


procedure Forecast (var observ, frcst, mad:quarterArray;
                var stepIndArry, trndIndArry,mkCodeArry: qtrIntArray;
                numberOfQtrs,repNum:integer; unitPrice:real);

const ALPHA = 0.1;
      STEPBOUND1 = 3.0;
      STEPBOUND2 = 2.0;

var upper, lower, sum, sampleMean, sampleStdDev, stdDevToMean:real;
    upInd, downInd, stepInd, trendInd, trendUp,
    trendDn, t, i, j, W, S, table:integer;
    kendTest, lowDemand:boolean;
```

```
begin
  writeln('Running Replication # ',repNum);
  mkCodeArry[1]:=getMarkCode (1,0,frcst[1],unitPrice);
  upInd:=0;downInd:=0;
  for t:=2 to numberOfQtrs do begin          {Compute quarterly forecast}
    lowDemand:=FALSE;
    trendInd:=0;
    stepInd:=0;
    if ((mkCodeArry[t-1] = 0) or (mkCodeArry[t-1] = 1) or (mkCodeArry[t-1] =3)) then
lowDemand:=TRUE;
    if lowDemand then begin
      upper:=STEPBOUND1*frcst[t-1];
      lower:=0.0;
    end else begin
      upper:=frcst[t-1]+1.25*mad[t-1]*STEPBOUND2;
      lower:=frcst[t-1]-1.25*mad[t-1]*STEPBOUND2;
    end;
    if (lowDemand and (observ[t-1] < 5)) or
      ((observ[t-1] < upper) and (observ[t-1] >= lower)) then begin
      upInd:=0;
      downInd:=0;
      frcst[t]:=ALPHA*observ[t-1]+(1-ALPHA)*frcst[t-1];
      mad[t]:=ALPHA*(abs(observ[t-1]-frcst[t-1]))+(1-ALPHA)*mad[t-1];
    end else begin
      if ((observ[t-1] > upper) and (upInd=1)) or
        ((observ[t-1] < lower) and (downInd=1)) then begin
        if t>4 then begin
          frcst[t]:=(observ[t-4]+observ[t-3]+observ[t-2]+observ[t-1])/4;
        end else if t = 4 then begin
          frcst[t]:=(observ[t-3]+observ[t-2]+observ[t-1])/3;
        end else if t = 3 then begin
          frcst[t]:=(observ[t-2]+observ[t-1])/2;
        end;
        if frcst[t] > ERROR then begin
          mad[t]:=COEFF1*exp(POWER1*ln(frcst[t]));
        end else begin
          mad[t]:=0.0;
        end;
        stepInd:=1;
        upInd:=0;
        downInd:=0;
      end else begin
        if ((observ[t-1] > upper) and (upInd=0)) then begin
          upInd:=1;
          frcst[t]:=frcst[t-1];
          mad[t]:=mad[t-1];
        end else begin
          if ((observ[t-1] < lower) and (downInd=0)) then begin
            downInd:=1;
            frcst[t]:=frcst[t-1];
```

```
          mad[t]:=mad[t-1];
        end;
      end;
    end;
  end;
  if (t>4) and (stepInd=0) then begin     {Conduct Kendall Trend Test}
    sum:=0.0;
    if t <= 8 then begin
      for i:=1 to t-1 do begin
        sum:=sum+observ[i];
      end;
      sampleMean:=sum/(t-1);
      sum:=0.0;
      for i:=1 to t-1 do begin
        sum:=sum+sqr(observ[i]-sampleMean);
      end;
      sampleStdDev:=sqrt(sum/(t-2));
    end else begin
      for i:=t-8 to t-1 do begin
        sum:=sum+observ[i];
      end;
      sampleMean:=sum/8;
      sum:=0.0;
      for i:=t-8 to t-1 do begin
        sum:=sum+sqr(observ[i]-sampleMean);
      end;
      sampleStdDev:=sqrt(sum/7);
    end;
    if sampleMean > 0.0 then begin
      stdDevToMean:=sampleStdDev/sampleMean
    end else begin
      stdDevToMean:=99999.0
    end;
    kendTest:=false;
    if (sampleMean >= 3.0) and (stdDevToMean <= 1.75) then begin
      kendTest:=true;
      if stdDevToMean > 1.0 then begin
        table:=3;
      end else begin
        table:=2;
      end;
    end;
    if ((sampleMean >= 1.0) and (sampleMean < 3.0)) and
      (stdDevToMean <= 1.75) then begin
      kendTest:=true;
      if stdDevToMean > 1.25 then begin
        table:=3;
      end else begin
        table:=2;
      end;
```

```
end;
if ((sampleMean >= 0.125) and (sampleMean < 1.0)) and
  (stdDevToMean <= 2.00) then begin
  kendTest:=true;
  table:=2;
end;
if kendTest=true then begin        {Conduct Kendall S-Test for Trend}
  W:=8;
  if (sampleMean >= 3.0) and (sampleMean < 9.0) then begin
    if (stdDevToMean < 0.30) then W:=6;
    end;
  if (sampleMean >= 9.0) and (sampleMean < 20.0) then begin
    if (stdDevToMean < 0.93) then W:=6;
    if (stdDevToMean < 0.28) then W:=4;
    end;
  if (sampleMean >= 20.0) then begin
    if (stdDevToMean < 0.53) then W:=6;
    if (stdDevToMean < 0.28) then W:=4;
    end;
  if W > (t-1) then W:=((t-1) div 2)*2;
  S:=0;
  for i:=(t-W) to (t-2) do begin      {Compute Kendall S-Statistic}
    for j:=(i+1) to (t-1) do begin
      if observ[i] < observ[j] then S:=S+1;
      if observ[i] > observ[j] then S:=S-1;
    end;
  end; {for}
  if table = 2 then begin
    if W = 4 then begin
      trendUp:=4; trendDn:=-4;
    end;
    if W = 6 then begin
      trendUp:=9; trendDn:=-9;
    end;
    if W = 8 then begin
      trendUp:=13; trendDn:=-13;
    end;
  end else begin
    if W = 4 then begin
      trendUp:=6; trendDn:=-6;
    end;
    if W = 6 then begin
      trendUp:=11; trendDn:=-11;
    end;
    if W = 8 then begin
      trendUp:=16; trendDn:=-16;
    end;
  end; {if}
  trendInd:=0;
  if S >= trendUp then trendInd:=1;
```

```
        if S <= trendDn then trendInd:=1;
        if trendInd =1 then begin
          sum:=0.0;
          for i:=(t-4) to (t-1) do begin
            sum:=sum+observ[i];
            end;
          frcst[t]:=sum/4;
          if frcst[t] > ERROR then begin
            mad[t]:=COEFF1*exp(POWER1*ln(frcst[t]));
          end else begin
            mad[t]:=0.0;
          end;
        end; {if}
      end; {if}
    end; {if}
  mkCodeArry[t]:=getMarkCode (t,mkCodeArry[t-1],frcst[t],unitPrice);
  stepIndArry[t]:=stepInd;
  trndIndArry[t]:=trendInd;
  end; {for}
end;


procedure SDR(var OSHeap,BOHeap:PriorityQueueType;
          var wklyObserv:weeklyArray;
          var frcst,meanDmdArry,observ,mad:quarterArray;
          var numberOfQtrs,initInv,initOS,initOrders,startSSQtr,endSSQtr,
            initSSOH,initSSOS,initSSOrders:integer;
          var meanDemand:real;
          var TWUS:longint;
          var ACWTBO,ACWT,SMA,Invest,orderCount,lastOH,lastOS,totalCost,
            inappAsset,inappVal:real;
          wkDataType,qtrDataType,outputType,silverSSType:char;
          unitPrice,PLT,P1,adminCost,obsol,timePref,storage,shortCost,
          frcstErrCoeff,bufferMult,PLTSigMuRatio:real;
          ROLowConst,maxQtrs,minQtrs,maxDecl:integer;
          var qtrACWTBOArry,qtrACWTArry,qtrSMAArry,qtrInvestArry,
          qtrInappArry,cumACWTBOArry,cumACWTArry,cumSMAArry:qtrStatArry;
          numberRep:integer);

var  wklyBO,wklyOS:datarecord;
    amtBO,amtRecv,receipt,wklyDemand,date,BOqtr,endQtr,
    T1Part1,T1Part2,lastT1Order:integer;
    t,wk,qtr,sizeOS,sizeBO,orderQty,initOrderQty:integer;
    randnorm,randPLT,wklyInvest,qtrInvest,repInvest,holdCost,cumSSHoldCost,
    twoYearAmt,qtrInapp,intLength,startInt,SSOrderCount:real;
    flag1,flag2:boolean;
    BOFill,dmdTot,SSOSTot,OSCurr,BOTot,BOCurr,OHcurr,IPcurr:integer;
    oldCumACWTBO,oldCumACWT,oldCumSMA,oldQtrInvest,oldQtrACWTBO,oldQtrInapp,
    oldQtrACWT,oldQtrSMA,ACWTBOvalue,ACWTvalue,SMAvalue:real;
    qtrTWUSArry,qtrBOTotArry,qtrBOFillArry:qtrIntArray;
```

```
procedure SilverModel (qtr,IPcurr,numberOfQtrs:integer;
                       holdCost,PLT,P1,adminCost,frcstErrCoeff,
                       bufferMult:real; ROLowConst,maxQtrs,minQtrs,maxDecl:integer;
                       var frcst,meanDmdArry,mad,observ:quarterArray;
                       var orderQty,lastT1Order,T1part1,T1Part2:integer;
                       wk:integer;
                       silverSSType:char);

var ka,kr,frcstTotal,cut,minCut,X1,X2,X3,sigmaX1,sigmaX2,sigmaX3,X1mu,X3mu,
   sumSqX1,sumSqX2,sumSqX3,C,B,D,PLTBuffer,partial1,partial2:real;
   i,j,T,endLTqtr,horizQtrs:integer;
   qtrFrcstArry:quarterArray;


function GetC(qtr,wk:integer;madVal,frcstVal:real;
          var frcst,observ:quarterArray):real;

const MADWGHT = 1.0;

var t:integer;
   sum,sumSq,mean:real;

begin
  if frcstVal > 1.0 then begin
    if wk = 13 then begin
      if qtr >= 8 then begin
        sum:=0.0;
        for t:=qtr-7 to qtr do begin
          sum:=sum+frcst[t];
        end;
        mean:=sum/8;
        sumSq:=0.0;
        for t:=qtr-7 to qtr do begin
          sumSq:=sumSq+sqr(observ[t]-mean);
        end;
        if (sumSq < ERROR) or (mean < ERROR) then begin
          GetC:=(1.25*madVal/frcstVal);
        end else begin
          GetC:=MADWGHT*(1.25*madVal/frcstVal)+
              (1-MADWGHT)*((sqrt(sumSq/7))/mean);
        end;
      end else begin
        GetC:=(1.25*madVal/frcstVal);
      end;
    end else begin
      if qtr > 8 then begin
        sum:=0.0;
        for t:=qtr-8 to (qtr-1) do begin
          sum:=sum+frcst[t];
        end;
```

```
          mean:=sum/8;
          sumSq:=0.0;
          for t:=qtr-8 to (qtr-1) do begin
            sumSq:=sumSq+sqr(observ[t]-mean);
          end;
          if (sumSq < ERROR) or (mean < ERROR) then begin
            GetC:=(1.25*madVal/frcstVal);
          end else begin
            GetC:=MADWGHT*(1.25*madVal/frcstVal)+
                (1-MADWGHT)*((sqrt(sumSq/7))/mean);
          end;
        end else begin
          GetC:=(1.25*madVal/frcstVal);
        end;
      end;
    end else begin
      GetC:=(1.25*madVal/frcstVal);
    end;
end; {GetC}


begin
  B:=bufferMult;
  C:=frcstErrCoeff;
  if wk=13 then begin
    if ((qtr+round(PLT)+1) <= numberOfQtrs) and (meanDmdArry[qtr+1] > ERROR) and
      (frcst[qtr+1] > ERROR) then begin
      endLTqtr:=qtr+round(PLT);
      horizQtrs:=maxQtrs;
      if (frcstErrCoeff=0.0) then begin                    {default}
        C:=GetC(qtr,wk,mad[qtr+1],frcst[qtr+1],frcst,observ);
      end;
      if (numberOfQtrs-endLTqtr) < maxQtrs then
        horizQtrs:=numberOfQtrs-endLTqtr;
      if silverSSType = '0' then begin
        for i:=(qtr+1) to (endLTqtr+horizQtrs) do begin
          qtrFrcstArry[i]:=(meanDmdArry[i]/meanDmdArry[qtr+1])*frcst[qtr+1];
        end;
        for i:=(endLTqtr+1) to (endLTQtr+horizQtrs) do begin
          if qtrFrcstArry[i] < qtrFrcstArry[endLTqtr] then horizQtrs:=maxDecl;
        end;
      end else begin
        for i:=(qtr+1) to (endLTqtr+horizQtrs) do begin
          qtrFrcstArry[i]:=frcst[qtr+1];
        end;
      end;
      X1:=0.0;
      sumSqX1:=0.0;
      if lastT1Order < 13 then begin
        T1Part1:=T1Part1-1;
```

```
  for i:=(qtr+1) to (qtr+round(PLT))  do begin
    X1:=X1  +  qtrFrcstArry[i];
    sumSqX1:=sumSqX1+sqr(qtrFrcstArry[i]);
  end;
  X1:=X1  +  (T1Part2/13)*qtrFrcstArry[qtr+round(PLT)+1];
  sumSqX1:=sumSqX1+sqr((T1Part2/13)*qtrFrcstArry[qtr+round(PLT)+1]);
  X1mu:=X1/(round(PLT)+(1-(lastT1Order/13)));
end else begin
  for i:=(qtr+1) to (qtr+round(PLT)+1)  do begin
    X1:=X1  +  qtrFrcstArry[i];
    sumSqX1:=sumSqX1+sqr(qtrFrcstArry[i]);
  end;
  X1mu:=X1/(round(PLT)+1);
end;
if X1 > ERROR then begin
  sigmaX1:=sqrt(sqr(C)*(sumSqX1)+sqr(X1mu)*sqr(PLTSigMuRatio*PLT));
  ka:=(IPcurr-X1)/sigmaX1;        { actual safety factor }
end;
kr:=ZInv(P1);                     { required safety factor }
if (ka < kr) and (X1>ERROR) then begin   { then place an order }
  T:=1;
  for i:=1 to horizQtrs do begin
    frcstTotal:=0.0;
    for j:=1 to i do begin
      frcstTotal:=frcstTotal+(j-1)*qtrFrcstArry[endLTqtr+i];
    end;
    cut:=(adminCost+holdCost*frcstTotal)/i;
    if i=1 then begin
      minCut:=cut;
    end else if cut < minCut then begin
      minCut:=cut;
      T:=i;
    end;
  end; {for}
  if T < minQtrs then T:=minQtrs;
  if T=1 then begin
    if lastT1Order >= 13 then begin
      orderQty:=round(X1+B*sqrt(sqr(C*qtrFrcstArry[qtr+1])/13)+
                kr*sigmaX1)-IPCurr;
      lastT1Order:=0;
      T1Part1:=1-wk;
      T1Part2:=wk;
    end else begin
      X1:=0.0;
      sumSqX1:=0.0;
      for i:=(qtr+1) to (qtr+round(PLT)+1)  do begin
        X1:=X1  +  qtrFrcstArry[i];
        sumSqX1:=sumSqX1+sqr(qtrFrcstArry[i]);
      end;
      X1mu:=X1/(round(PLT)+1);
```

```
            if X1 > ERROR then
               sigmaX1:=sqrt(sqr(C)*(sumSqX1)+sqr(X1mu)*sqr(PLTSigMuRatio*PLT));
            orderQty:=round(X1+B*sqrt(sqr(C*qtrFrcstArry[qtr+1])/13)+
                        kr*sigmaX1)-IPCurr;
            lastT1Order:=0;
            T1Part1:=1-wk;
            T1Part2:=wk;
          end;
        end else begin
          lastT1Order:=13;
          T1Part1:=0;
          T1Part2:=0;
          X2:=0.0;
          sumSqX2:=0.0;
          for i:=(qtr+1) to (endLTqtr+T-1-round(PLT)) do begin
            X2:=X2 + qtrFrcstArry [i];
            sumSqX2:=sumSqX2+sqr(qtrFrcstArry[i]);
          end;
          sigmaX2:=C*sqrt(sumSqX2);
          X3:=0.0;
          sumSqX3:=0.0;
          for i:=(endLTqtr+T-round(PLT)) to (endLTqtr+T) do begin
            X3:=X3 + qtrFrcstArry [i];
            sumSqX3:=sumSqX3+sqr(qtrFrcstArry[i]);
          end;
          X3mu:=X3/(round(PLT)+1);
          sigmaX3:=sqrt(sqr(C)*(sumSqX3)+sqr(X3mu)*sqr(PLTSigMuRatio*PLT));
          orderQty:=round(X3+kr*sigmaX3+B*sigmaX2+X2-IPcurr);
        end;
        if orderQty < ROLowConst then orderQty:=ROLowConst;
      end; {if}
    end; {if}
  end else begin
    partial1:=1-wk/13;
    partial2:=wk/13;
    if ((qtr-partial1+round(PLT)+1) <= numberOfQtrs) and (meanDmdArry[qtr] > ERROR)
and
      (frcst[qtr] > ERROR) then begin
      endLTqtr:=qtr+round(PLT);
      horizQtrs:=maxQtrs;
      if (frcstErrCoeff=0.0) then begin
        C:=GetC(qtr,wk,mad[qtr],frcst[qtr],frcst,observ);
      end;
      if (numberOfQtrs-endLTqtr) < maxQtrs then
        horizQtrs:=numberOfQtrs-endLTqtr;
      if silverSSType = '0' then begin
        for i:=qtr to (endLTqtr+horizQtrs) do begin
          qtrFrcstArry[i]:=(meanDmdArry[i]/meanDmdArry[qtr])*frcst[qtr];
        end;
        for i:=(endLTqtr+1) to (endLTQtr+horizQtrs) do begin
```

```
        if qtrFrcstArry[i] < qtrFrcstArry[endLTqtr] then horizQtrs:=maxDecl;
      end;
    end else begin
      for i:=qtr to (endLTqtr+horizQtrs) do begin
        qtrFrcstArry[i]:=frcst[qtr];
      end;
    end;
    X1:=0.0;
    sumSqX1:=0.0;
    if lastT1Order < 13 then begin
      T1Part1:=T1Part1-1;
      if T1Part1 > 0 then begin
        for i:=(qtr) to (qtr+round(PLT)-1) do begin
          X1:=X1 + (partial1*qtrFrcstArry[i]+partial2*qtrFrcstArry[i+1]);
          sumSqX1:=sumSqX1+sqr(partial1*qtrFrcstArry[i]+
                    partial2*qtrFrcstArry[i+1]);
        end;
        X1:=X1 + partial1*qtrFrcstArry[qtr+round(PLT)] +
              ((T1Part2)/13)*qtrFrcstArry[qtr+round(PLT)+1];
        sumSqX1:=sumSqX1 + sqr(partial1*qtrFrcstArry[qtr+round(PLT)] +
                  ((T1Part2)/13)*qtrFrcstArry[qtr+round(PLT)+1]);
        X1mu:=X1/(round(PLT)+(1-(lastT1Order/13)));
      end else begin
        for i:=(qtr) to (qtr+round(PLT)-1) do begin
          X1:=X1 + (partial1*qtrFrcstArry[i]+partial2*qtrFrcstArry[i+1]);
          sumSqX1:=sumSqX1+sqr(partial1*qtrFrcstArry[i]+
                    partial2*qtrFrcstArry[i+1]);
        end;
        X1:=X1 + ((T1Part2+T1Part1)/13)*qtrFrcstArry[qtr+round(PLT)];
        sumSqX1:=sumSqX1 +
sqr(((T1Part2+T1Part1)/13)*qtrFrcstArry[qtr+round(PLT)]);
        X1mu:=X1/(round(PLT)+(1-(lastT1Order/13)));
      end;
    end else begin
      for i:=(qtr) to (qtr+round(PLT)) do begin
        X1:=X1 + (partial1*qtrFrcstArry[i]+partial2*qtrFrcstArry[i+1]);
        sumSqX1:=sumSqX1+sqr(partial1*qtrFrcstArry[i]+
                  partial2*qtrFrcstArry[i+1]);
      end;
      X1mu:=X1/(round(PLT)+1);
    end;
    if X1 > ERROR then begin
      sigmaX1:=sqrt(sqr(C)*(sumSqX1)+sqr(X1mu)*sqr(PLTSigMuRatio*PLT));
      ka:=(IPcurr-X1)/sigmaX1;        { actual safety factor }
    end;
    kr:=ZInv(P1);                     { required safety factor }
    if (ka < kr) and (X1>ERROR) then begin  { then place an order }
      T:=1;
      for i:=1 to horizQtrs do begin
        frcstTotal:=0.0;
```

```
    for j:=1 to i do begin
      frcstTotal:=frcstTotal+(j-1)*(partial1*qtrFrcstArry[endLTqtr+i-1]+
                                    partial2*qtrFrcstArry[endLTqtr+i]);
    end;
    cut:=(adminCost+holdCost*frcstTotal)/i;
    if i=1 then begin
      minCut:=cut;
    end else if cut < minCut then begin
      minCut:=cut;
      T:=i;
    end;
  end; {for}
  if T < minQtrs then T:=minQtrs;
  if T=1 then begin
    if lastT1Order >= 13 then begin
      orderQty:=round(X1+B*sqrt(sqr(C*qtrFrcstArry[qtr])/13)+
                      kr*sigmaX1)-IPCurr;
      lastT1Order:=0;
      T1Part1:=1-wk;
      T1Part2:=wk;
    end else begin
      X1:=0.0;
      sumSqX1:=0.0;
      for i:=(qtr) to (qtr+round(PLT)) do begin
        X1:=X1 + (partial1*qtrFrcstArry[i]+partial2*qtrFrcstArry[i+1]);
        sumSqX1:=sumSqX1+sqr(partial1*qtrFrcstArry[i]+
                             partial2*qtrFrcstArry[i+1]);
      end;
      X1mu:=X1/(round(PLT)+1);
      if X1 > ERROR then
        sigmaX1:=sqrt(sqr(C)*(sumSqX1)+sqr(X1mu)*sqr(PLTSigMuRatio*PLT));
      orderQty:=round(X1+B*sqrt(sqr(C*qtrFrcstArry[qtr])/13)+
                      kr*sigmaX1)-IPCurr;
      lastT1Order:=0;
      T1Part1:=1-wk;
      T1Part2:=wk;
    end;
  end else begin
    lastT1Order:=13;
    T1Part1:=0;
    T1Part2:=0;
    X2:=0.0;
    sumSqX2:=0.0;
    for i:=qtr to (endLTqtr+T-2-round(PLT)) do begin
      X2:=X2 + (partial1*qtrFrcstArry[i]+partial2*qtrFrcstArry[i+1]);
      sumSqX2:=sumSqX2 + sqr(partial1*qtrFrcstArry[i]+
                             partial2*qtrFrcstArry[i+1]);
    end;
    sigmaX2:=C*sqrt(sumSqX2);
    X3:=0.0;
```

```
            sumSqX3:=0.0;
            for i:=(endLTqtr+T-1-round(PLT)) to (endLTqtr+T-1)  do begin
               X3:=X3 + (partial1*qtrFrcstArry[i]+partial2*qtrFrcstArry[i+1]);
               sumSqX3:=sumSqX3 + sqr(partial1*qtrFrcstArry[i]+
                              partial2*qtrFrcstArry[i+1]);
            end;
            X3mu:=X3/(round(PLT)+1);
            sigmaX3:=sqrt(sqr(C)*(sumSqX3)+sqr(X3mu)*sqr(PLTSigMuRatio*PLT));
            orderQty:=round(X3+kr*sigmaX3+B*sigmaX2+X2-IPcurr);
          end;
          if orderQty < ROLowConst then orderQty:=ROLowConst;
        end; {if}
      end;  {if}
    end; {if}
end; {SilverModel}



begin
  holdCost:=unitPrice*(obsol+timePref+storage)/4;
  InitializePriorityQueue(OSHeap);InitializePriorityQueue(BOHeap);
  OHCurr:=initInv;
  initOrderQty:=0;
  if initOS=0 then initOrders:=0;
  if (initOrders > 0) then begin
    initOrderQty:=initOS div initOrders;
    intLength:= (13*PLT)/initOrders;
    startInt:=0.0;
    for i:=1 to initOrders do begin
      wklyOS.Qty:=initOrderQty;
      wklyOS.Week:=round((startInt + (i*intLength))/2);
      InsertPriorityQueue(OSHeap,wklyOS);
      startInt:=startInt+intLength;
    end;
  end;
  initOS:=initOrderQty*initOrders;
  OSCurr:=initOS;
  IPCurr:=OHCurr + OSCurr;

  if (qtrDataType = '1') or (wkDataType = '1') then begin
    writeln(outputfile);
    writeln(outputfile,'SDR Data        Initial OH Inv:= ',initInv);
    writeln(outputfile,'-------------------------------------------------------------------------');
  end;

  for t:=1 to numberOfQtrs do begin
    qtrTWUSArry[t]:=0;
    qtrBOTotArry[t]:=0;
```

```
  qtrBOFillArry[t]:=0;
end;
BOCurr:=0;
replnvest:= 0.0;
date:=1;
lastT1Order:=13;
T1Part1:=0;
T1part2:=0;

for qtr:= 1 to numberOfQtrs do begin

  if (qtr >= startSSQtr) and (qtr <= endSSQtr) then begin
    if wkDataType = '1' then begin
      writeln(outputfile);
      writeln(outputfile,'QTR WK   REC  DEM   BO  OS   OH  IP ORDCNT');
    end;
   end;

  if qtr=startSSQtr then begin
    initSSOH:=OHCurr;
    initSSOS:=OSCurr;
    initSSOrders:=SizePriorityQueue(OSHeap);
    cumSSHoldCost:=0.0;
    SSOSTot:=0;
    SSOrderCount:=0.0;
  end;
  qtrInvest:= 0.0;
  wklyInvest:= 0.0;

  for wk:= 1 to 13 do begin
    wklyDemand:=round(wklyObserv[date]);
    receipt:=0;
    amtRecv:=0;
    amtBO:=0;
    wklyBO.Qty:=0;
    wklyBO.Week:=date;
    wklyOS.Qty:=0;
    flag1:=FALSE; flag2:=FALSE;

    if not (EmptyPriorityQueue(OSHeap)) then begin          {receive}
      repeat
        if CurrWeek(OSHeap) = date then begin
          amtRecv:=ExtractQty(OSHeap);
          receipt:=amtRecv;
          OSCurr:= OSCurr - amtRecv;
          while (amtRecv > 0) and not (EmptyPriorityQueue(BOHeap)) do begin
            if CurrQty(BOHeap) <= amtRecv then begin
              amtBO:=CurrQty(BOHeap);
              amtRecv:= amtRecv - amtBO;
              BOCurr:= BOCurr - amtBO;
```

```
                if (CurrWeek(BOHeap) mod 13) = 0 then begin
                  BOqtr:=(CurrWeek(BOHeap) div 13);
                end else begin
                  BOqtr:=(CurrWeek(BOHeap) div 13)+1;
                end;
                qtrBOFillArry[BOqtr]:=qtrBOFillArry[BOqtr]+amtBO;
                qtrTWUSArry[BOqtr]:=qtrTWUSArry[BOqtr]+(amtBo*(date -
ExtractWeek(BOHeap)));
              end else begin
                BOHeap.HeapArray[1].Qty:= BOHeap.HeapArray[1].Qty - amtRecv;
                if (BOHeap.HeapArray[1].Week mod 13) = 0 then begin
                  BOqtr:=(BOHeap.HeapArray[1].Week) div 13;
                end else begin
                  BOqtr:=((BOHeap.HeapArray[1].Week) div 13)+1;
                end;
                qtrTWUSArry[BOqtr]:=qtrTWUSArry[BOqtr]+(amtRecv*(date -
BOHeap.HeapArray[1].Week));
                BOCurr:= BOCurr - amtRecv;
                qtrBOFillArry[BOqtr]:=qtrBOFillArry[BOqtr]+amtRecv;
                amtRecv:= 0;
              end; {if}
            end; {while}
            OHCurr:=OHCurr + amtRecv;
          end;
          if EmptyPriorityQueue(OSHeap) then flag2:= TRUE
          else if currWeek(OSHeap) <> date then flag1:=TRUE;
        until flag1 or flag2;
      end; {if  receive}

      if wklyDemand > 0 then begin              {issue}
        if wklyDemand > OHCurr then begin
          wklyBO.Qty:= wklyDemand - OHCurr;
          OHCurr:=0;
          InsertPriorityQueue(BOHeap,wklyBO);
          qtrBOTotArry[qtr]:=qtrBOTotArry[qtr]+wklyBO.Qty;
          BOCurr:=BOCurr + wklyBO.Qty;
        end else begin
          OHCurr:= OHCurr - wklyDemand;
        end;
      end; {if  issue}


      orderQty:=0;                          {order}
      IPCurr:= OHCurr + OSCurr - BOCurr;
      lastT1Order:=lastT1Order+1;
      SilverModel (qtr,IPcurr,numberOfQtrs,
              holdCost,PLT,P1,adminCost,frcstErrCoeff,
              bufferMult,ROLowConst,maxQtrs,minQtrs,maxDecl,
              frcst,meanDmdArry,mad,observ,orderQty,
              lastT1Order,T1Part1,T1Part2,wk,
```

```
            silverSSType);
  if orderQty>0 then begin
    wklyOS.Qty:=orderQty;
    randnorm:=GetNormal;
    randPLT:=abs(PLT+(randnorm*PLTSigMuRatio*PLT));
    if randPLT > MAXPLT then begin
      randPLT:=MAXPLT;
    end else if randPLT < MINPLT then begin
      randPLT:=MINPLT
    end;
    wklyOS.Week:=date + round(randPLT*13) + 1;
    InsertPriorityQueue(OSHeap,wklyOS);
    OSCurr:= OSCurr + wklyOS.Qty;
    if (qtr >= startSSQtr) and (qtr <= endSSQtr) then begin
      SSOrderCount:=SSOrderCount + 1.0;
      SSOSTot:= SSOSTot + wklyOS.Qty;
    end;
  end; {if}

  if (qtr >= startSSQtr) and (qtr <= endSSQtr) then begin
    if wkDataType = '1' then begin
      writeln(outputfile,qtr:3,date:5,receipt:6,wklyDemand:6,BOCurr:6,
            OSCurr:6,OHCurr:6,IPCurr:6,SSOrderCount:6:0);
      if (outputType = '1') and ((wk mod 13) = 0) then HitToCont;
    end;
   end;

  receipt:=0;
  date:=date+1;
  wklyInvest:= wklyInvest + OSCurr + OHCurr;
  if (qtr >= startSSQtr) and (qtr <= endSSQtr) then begin
    cumSSHoldCost:=cumSSHoldCost+OHCurr*holdCost/13;
  end;
end; {for week}

qtrInvest:= wklyInvest/13;
if (qtr >= startSSQtr) and (qtr <= endSSQtr) then begin
   repInvest:= repInvest + qtrInvest;
end;

oldQtrInvest:=qtrInvestArry[qtr].Mean;
qtrInvestArry[qtr].Mean:=NewMean(qtrInvestArry[qtr].Mean,qtrInvest,numberRep);
qtrInvestArry[qtr].Variance:=NewVar(qtrInvestArry[qtr].Mean,oldQtrInvest,
                        qtrInvestArry[qtr].Variance,qtrInvest,numberRep);

twoYearAmt:=0.0;
endQtr:=qtr+8;
if qtr<numberOfQtrs then begin
  if (numberOfQtrs-qtr) < 8 then begin
    endQtr:=numberOfQtrs;
```

```
          if silverSSType = '0' then begin
            if meanDmdArry[qtr + 1] > 0.0 then begin
              for i:= (qtr + 1) to (endQtr) do begin

twoYearAmt:= twoYearAmt + (meanDmdArry[i]/meanDmdArry[qtr + 1])*frcst[qtr + 1];
              end;
              twoYearAmt:= twoYearAmt +

(8-(numberOfQtrs-qtr))*(meanDmdArry[endQtr]/meanDmdArry[qtr + 1])*frcst[qtr + 1];
            end;
          end else begin
            twoYearAmt:= 8*frcst[qtr + 1];
          end;
        end else begin
          if silverSSType = '0' then begin
            if meanDmdArry[qtr + 1] > 0.0 then begin
              for i:= (qtr + 1) to (qtr + 8) do begin

twoYearAmt:= twoYearAmt + (meanDmdArry[i]/meanDmdArry[qtr + 1])*frcst[qtr + 1];
              end;
            end;
          end else begin
            twoYearAmt:= 8*frcst[qtr + 1];
          end;
        end;
      end;
      qtrInapp:= OHCurr-twoYearAmt;
      if qtrInapp < 0.0 then qtrInapp:= 0.0;
      oldQtrInapp:= qtrInappArry[qtr].Mean;
      qtrInappArry[qtr].Mean:= NewMean(qtrInappArry[qtr].Mean,qtrInapp,numberRep);
      qtrInappArry[qtr].Variance:= NewVar(qtrInappArry[qtr].Mean,oldQtrInapp,
                              qtrInappArry[qtr].Variance,qtrInapp,numberRep);

      if qtr = endSSQtr then begin
        invest:= repInvest/(endSSQtr-startSSQtr + 1);
        lastOH:= OHCurr;
        lastOS:= OSCurr;
        inappAsset:= qtrInapp;
        inappVal:= qtrInapp*unitPrice;
        orderCount:= SSOrderCount;
      end;

      if (qtr >= startSSQtr) and (qtr <= endSSQtr) then begin
        if (wkDataType='1') then begin
          writeln(outputfile);
          writeln(outputfile,'QTR   DMD    OH    IP    OS    BO   INVEST');
        end else  if qtrDataType = '1' then begin
          if (qtr=1) or (((qtr-1) mod 20 ) = 0)then begin
            writeln(outputfile);
            writeln(outputfile,'QTR   DMD    OH    IP    OS    BO  INVEST ');
```

```
        end;
      end;
      if qtrDataType = '1' then
        writeln(outputfile,qtr:3,observ[qtr]:6:0,OHCurr:6,IPCurr:6,
        OSCurr:6,BOCurr:6,qtrInvest:8:2);
      if (outputType = '1') and (qtrDataType ='1') and (((qtr-1) mod 20) = 0) then
HitToCont;
    end;

  end; {for qtr}

  if not (EmptyPriorityQueue(OSHeap)) then begin    {adjust final qtr TWUS}
    while not (EmptyPriorityQueue(BOHeap)) do begin
      amtBO:=CurrQty(BOHeap);
      if (CurrWeek(BOHeap) mod 13) = 0 then begin
        BOqtr:=(CurrWeek(BOHeap) div 13);
      end else begin
        BOqtr:=(CurrWeek(BOHeap) div 13)+1;
      end;
      qtrBOFillArry[BOqtr]:=qtrBOFillArry[BOqtr]+amtBO;
      qtrTWUSArry[BOqtr]:=qtrTWUSArry[BOqtr]+
                  (amtBo*(OSHeap.HeapArray[1].Week - ExtractWeek(BOHeap)));
    end; {while}
  end;

  for t:= 1 to numberOfQtrs do begin
    if qtrBOFillArry[t] > 0 then begin
      oldQtrACWTBO:=qtrACWTBOArry[t].Mean;
      ACWTBOvalue:=(7*(qtrTWUSArry[t]/qtrBOFillArry[t]));
      qtrACWTBOArry[t].Mean:=NewMean(qtrACWTBOArry[t].Mean,
                        ACWTBOvalue,numberRep);
      qtrACWTBOArry[t].Variance:=NewVar(qtrACWTBOArry[t].Mean,oldQtrACWTBO,
                        qtrACWTBOArry[t].Variance,ACWTBOvalue,numberRep);
    end; {if}
    oldQtrACWT:=qtrACWTArry[t].Mean;
    if observ[t] > 0 then begin
      ACWTvalue:=(7*(qtrTWUSArry[t]/observ[t]));
    end else begin
      ACWTvalue:=0.0;
    end;
    qtrACWTArry[t].Mean:=NewMean(qtrACWTArry[t].Mean,
                        ACWTvalue,numberRep);
    qtrACWTArry[t].Variance:=NewVar(qtrACWTArry[t].Mean,oldQtrACWT,
                        qtrACWTArry[t].Variance,ACWTvalue,numberRep);
    oldQtrSMA:=qtrSMAArry[t].Mean;
    if observ[t] > 0 then begin
      SMAvalue:= (1-(qtrBOTotArry[t]/observ[t]));
    end else begin
      SMAvalue:=1.0;
    end;
```

```
    qtrSMAArry[t].Mean:=NewMean(qtrSMAArry[t].Mean,
                          SMAvalue,numberRep);
    qtrSMAArry[t].Variance:=NewVar(qtrSMAArry[t].Mean,oldQtrSMA,
                         qtrSMAArry[t].Variance,SMAvalue,numberRep);
  end; {for}

  dmdTot:=0;
  TWUS:=0;
  BOTot:=0;
  BOFill:=0;

  for qtr:=startSSQtr to endSSQtr do begin

    dmdTot:=dmdTot + round(observ[qtr]);
    TWUS:=TWUS + qtrTWUSArry[qtr];
    BOTot:=BOTot + qtrBOTotArry[qtr];
    BOFill:= BOFILL + qtrBOFillArry[qtr];

    if BOFill <> 0 then begin
      ACWTBO:=7*(TWUS/BOFill);
    end else begin
      ACWTBO:=0.0;
    end; {if}
    if dmdTot <> 0 then begin
      ACWT:=7*(TWUS/dmdTot);
      SMA:=1 - BOTot/dmdTot;
    end else begin
      ACWT:=0.0;
      SMA:=1.0;
    end; {if}

    oldCumACWTBO:=cumACWTBOArry[qtr].Mean;

cumACWTBOArry[qtr].Mean:=NewMean(cumACWTBOArry[qtr].Mean,ACWTBO,numberRep
);

cumACWTBOArry[qtr].Variance:=NewVar(cumACWTBOArry[qtr].Mean,oldCumACWTBO,
                         cumACWTBOArry[qtr].Variance,ACWTBO,numberRep);
    oldCumACWT:=cumACWTArry[qtr].Mean;
    cumACWTArry[qtr].Mean:=NewMean(cumACWTArry[qtr].Mean,ACWT,numberRep);
    cumACWTArry[qtr].Variance:=NewVar(cumACWTArry[qtr].Mean,oldCumACWT,
                         cumACWTArry[qtr].Variance,ACWT,numberRep);
    oldCumSMA:=cumSMAArry[qtr].Mean;
    cumSMAArry[qtr].Mean:=NewMean(cumSMAArry[qtr].Mean,SMA,numberRep);
    cumSMAArry[qtr].Variance:=NewVar(cumSMAArry[qtr].Mean,oldCumSMA,
                         cumSMAArry[qtr].Variance,SMA,numberRep);
  end;  {for}

  totalCost:=(initSSOH+initSSOS+SSOSTot)*unitPrice+SSOrderCount*adminCost+
        cumSSHoldCost+(TWUS/52*shortCost);
```

```
end; {SDR}


procedure PrintHeader(unitPrice,PLT,P1,adminCost,obsol,timePref,storage,
                shortCost,frcstErrCoeff,bufferMult,
                PLTSigMuRatio,meanDemand, varDemand,negBinP:real;
                ROLowConst,maxQtrs,minQtrs,maxDecl,negBinS,seedIndex,
                initInv,initOS,initOrders:integer;
                var outputfile:text;outputType,distrType,silverSSType:char;
                outFileName:string;runDescript:descriptType;
                nmbrSteps,nmbrTrends,startSSQtr,endSSQtr:integer;
                stepMult,trendCoeff,trendPower:changeRealArry;
                startStep,starTrnd,endTrnd:changeIntArry);

var i:integer;
   distrUsed:string[7];
   Year,Month,Day,Dayofweek:word;

begin
  distrUsed:=' Normal';
  if distrType = '2' then distrUsed:='Poisson';
  if distrType = '3' then distrUsed:='Neg Binomial';
  if outputType = '2' then begin
    writeln(outputfile,'                    *** ',outFileName,' ***');
    writeln(outputfile);
    GetDate(Year,Month,Day,Dayofweek);
    if silverSSType = '0' then begin
      writeln(outputfile,' Date: ',Month,'-',Day,'-',Year,'    Model:  MOD SILVER
(VARIABLE FORECASTS)');
    end else begin
      writeln(outputfile,' Date: ',Month,'-',Day,'-',Year,'    Model:  MOD SILVER (FIXED
FORECASTS)');
    end;
  end;
  writeln(outputfile);
  writeln(outputfile,' Description: ',runDescript);
  writeln(outputfile);
  writeln(outputfile,' Initial simulation settings  ');
  writeln(outputfile);
  writeln(outputfile,'  Random number generator seed type:            ',seedtype);
  if seedType ='1' then begin
    writeln(outputfile,'  Random number seed start index:          ',seedIndex:6);
  end;
  writeln(outputfile,'  Type of demand distribution:             ', distrUsed);
  if distrType = '3' then begin
    writeln(outputfile,'  Neg Binomial Parameters:                p = ',negBinP:6:2);
    writeln(outputfile,'                                        s = ',negBinS:6);
  end;
  writeln(outputfile,'  Mean Demand:                              ',meanDemand:6:2);
  writeln(outputfile,'  Var Demand:                               ',varDemand:6:2);
```

```
  writeln(outputfile,'  Number of quarters to simulate:            ',numberOfQtrs:5 );
  writeln(outputfile,'  Start Sim Steady State quarter:           ',startSSQtr:5 );
  writeln(outputfile,'  End Sim Steady State quarter:             ',endSSQtr:5 );
  writeln(outputfile,'  Number of replications of simulation to run: ',numberOfReps:5);
  writeln(outputfile,'  Number of steps:                          ',nmbrSteps:5);

   if nmbrSteps >0 then begin
     for i:=1 to nmbrSteps do begin
       writeln(outputfile,'     Step: ',i:2,' Step Qtr: ',startStep[i]:4,
            ' Mult: ',stepMult[i]:7:4);
     end;
   end; {if}
   writeln(outputfile,'  Number of trends:                         ',nmbrTrends:5);
   if nmbrTrends >0 then begin
     for i:=1 to nmbrTrends do begin
        writeln(outputfile,'     Trend:',i:2,' Start Qtr: ',starTrnd[i]:3,
            ' Stop Qtr: ',endTrnd[i]:3,
            ' Coeff: ',trendCoeff[i]:7:4,' Power: ',trendPower[i]:7:4);
     end;
   end; {if}
   writeln(outputfile);
   if outputType = '1' then begin
     HitToCont;
     clrscr;
   end;
   writeln(outputfile,' Initial parameter settings  ');
   writeln (outputfile,' A.  Unit Price     : ',unitPrice:8:2,'    J.  Admin Order   :
',adminCost:8:2);
   writeln (outputfile,' B.  Buffer Mult (B): ',bufferMult:8:2,'    K.  R/O Constr    :
',ROLowConst:8);
   writeln (outputfile,' C.  Frcst Error (C): ',frcstErrCoeff:8:2,'    L.  Obsol Rate    :
',obsol:8:2);
   writeln (outputfile,' D.  PLT Sig/Mu     : ',PLTSigMuRatio:8:2,'    M.  Time Pref Rate:
',timePref:8:2);
   writeln (outputfile,' E.  Max Qtrs       : ',maxQtrs:8,'    N.  Storage Rate  : ',storage:8:2);
   writeln (outputfile,' F.  Max Decl Qtrs  : ',maxDecl:8,'    O.  Shortage Cost :
',shortCost:8:2);
   writeln (outputfile,' G.  Min Qtrs       : ',minQtrs:8,'    P.  Init Inv OH   : ',initInv:8);
   writeln (outputfile,' H.  Procur LT      : ',PLT:8:2,'    Q.  Init Qty OS   : ',initOS:8);
   writeln (outputfile,' I.  Risk           : ',P1:8:2,'    R.  Init Num Order: ',initOrders:8);

   if outputType = '1' then begin
     HitToCont;
     clrscr;
   end;
end;    {printheader}


procedure DisplayFrcstOutput (var  observ, frcst, mad :quarterArray;
```

```
                    var  stepIndArry, trndIndArry,mkCodeArry:qtrIntArray;
                    numberOfQtrs,initInv,repNum:integer;
                    outputType:char);

var t:integer;

begin
  writeln (outputfile);
  writeln(outputfile,'Replication Number ',repNum);
  writeln(outputfile);
  writeln(outputfile,'Quarterly Forecast Data');
  writeln(outputfile,'-------------------------------------------------------------------');
  for t:=1 to numberOfQtrs do begin
    if (t=1) or (((t-1) mod 20) = 0)then begin
      if (outputType='1') and (t>1) then HitToCont;
      writeln(outputfile);
      writeln (outputfile,'QTR   OBS  FRCST   MAD   MK  ST TR');
    end;
    writeln (outputfile,t:3,observ[t]:6:0,frcst[t]:8:2,mad[t]:8:2,
          mkCodeArry[t]:3, stepIndArry[t]:3,trndIndArry[t]:3);
  end;
  writeln (outputfile);
  if outputType= '1' then HitToCont;
end;

procedure DisplayRepStats (var ACWTBO, ACWT, SMA, Invest,orderCount,lastOH,
                    lastOS,totalCost,inappAsset,inappVal:real;
                    outputType:char );
begin
  if (numberRep = 1) or (outputType = '1') then begin
    writeln(outputfile);
    writeln
(outputfile,'*****************************************************************
*********');
    writeln(outputfile,' Rep# ACWTBO   ACWT    SMA  INVEST  End OH    Tot Cost
Inapp');
  end;
  writeln(outputfile,numberRep:5,ACWTBO:7:2,ACWT:7:2,SMA:7:2,Invest:8:2,lastOH:8:0,'
 ',totalCost:10:2,inappAsset:8:0);
  if outputType ='1' then begin
    delay(1500);
    clrscr;
  end;
end;


{procedure DisplayRepStats (var ACWTBO, ACWT, SMA, Invest,orderCount,lastOH,
                    lastOS,totalCost,inappAsset,inappVal:real;
                    outputType:char );
begin
```

```
  writeln(outputfile);
  writeln
(outputfile,'*****************************************************************
************');
  writeln(outputfile,'Replication # ',numberRep:3,' Final Statistics');
  writeln(outputfile,' ACWTBO   ACWT   SMA   Orders  INVEST   End OH  End OS');

writeln(outputfile,ACWTBO:7:2,ACWT:7:2,SMA:7:2,orderCount:8:0,Invest:8:2,lastOH:8:0,l
astOS:8:0);
  writeln(outputfile,' Total Cost  INAPP   INAPP Value');
  writeln(outputfile,totalCost:10:2,inappAsset:8:0,inappVal:10:2);

writeln(outputfile,'*****************************************************************
******************');
  if outputType ='1' then begin
    delay(1500);
    clrscr;
  end;
end;}


procedure CalcSimStats(ACWTBO, ACWT, SMA, Invest,orderCount,lastOH,lastOS,
                  totalCost,inappAsset,inappVal, initSSOH, initSSOS,
                  initSSOrders:real;
                  var n:integer;
                  var simACWTBO,simACWT,simSMA,simInvest,
                     simOrderCount,simLastOH,simLastOS,
                     simTotalCost,simACWTBOVar,simACWTVar,
                     simSMAVar,simInvestVar,simOrderCountVar,
                     simLastOHVar,simLastOSVar,
                     simTotalCostVar,simInapp,
                     simInappVar,simInappVal,simInappValVar,
                     simInitSSOH,simInitSSOS,simInitSSOrders,
                     simInitSSOHVar,simInitSSOSVar,
                     simInitSSOrdersVar:real);

var oldSimACWTBO,oldSimACWT,oldSimSMA,oldSimInvest,oldSimOrderCount,
   oldSimLastOH,oldSimLastOS,oldSimTotalCost,oldSimInapp,oldSimInappVal,
   oldSimInitSSOH,oldSimInitSSOS,oldSimInitSSOrders:real;

begin
  if n=0 then begin
    simACWTBO:=0.0;simACWT:=0.0;simSMA:=0.0;simInvest:=0.0;
    simOrderCount:=0.0;simLastOH:=0.0;simLastOS:=0.0;simTotalCost:=0;
    simInapp:=0.0;simInappVal:=0.0;
    simACWTBOVar:=0.0;simACWTVar:=0.0;simSMAVar:=0.0;
    simInvestVar:=0.0;simOrderCountVar:=0.0;simLastOHVar:=0.0;
    simLastOSVar:=0.0;simTotalCostVar:=0.0;
    simInappVar:=0.0;simInappValVar:=0.0;
    simInitSSOH:=0.0;simInitSSOS:=0.0;simInitSSOrders:=0.0;
```

```
    simInitSSOHVar:=0.0;simInitSSOSVar:=0.0;simInitSSOrdersVar:=0.0;
  end;

  n:=n+1;
  oldSimACWTBO:=simACWTBO;oldSimACWT:=simACWT;oldSimSMA:=simSMA;
  oldSimInvest:=simInvest;oldSimOrderCount:=simOrderCount;
  oldSimLastOH:=simLastOH;oldSimLastOS:=simLastOS;
  oldSimTotalCost:=simTotalCost;oldSimInapp:=simInapp;
  oldSimInappVal:=simInappVal;oldSimInitSSOH:=simInitSSOH;
  oldSimInitSSOS:=simInitSSOS;oldSimInitSSOrders:=simInitSSOrders;

  simACWTBO:=NewMean(simACWTBO,ACWTBO,n);
  simACWT:=NewMean(simACWT,ACWT,n);
  simSMA:=NewMean(simSMA,SMA,n);
  simInvest:=NewMean(simInvest,Invest,n);
  simOrderCount:=NewMean(simOrderCount,orderCount,n);
  simLastOH:=NewMean(simLastOH,lastOH,n);
  simLastOS:=NewMean(simLastOS,lastOS,n);
  simTotalCost:=NewMean(simTotalCost,totalCost,n);
  simInapp:=NewMean(simInapp,inappAsset,n);
  simInappVal:=NewMean(simInappVal,inappVal,n);
  simInitSSOH:=NewMean(simInitSSOH,initSSOH,n);
  simInitSSOS:=NewMean(simInitSSOS,initSSOS,n);
  simInitSSOrders:=NewMean(simInitSSOrders,initSSOrders,n);

  simACWTBOVar:=NewVar(simACWTBO,oldSimACWTBO,simACWTBOVar,ACWTBO,n);
  simACWTVar:=NewVar(simACWT,oldSimACWT,simACWTVar,ACWT,n);
  simSMAVar:=NewVar(simSMA,oldSimSMA,simSMAVar,SMA,n);
  simInvestVar:=NewVar(simInvest,oldSimInvest,simInvestVar,Invest,n);

simOrderCountVar:=NewVar(simOrderCount,oldSimOrderCount,simOrderCountVar,orderCou
nt,n);
  simlastOHVar:=NewVar(simLastOH,oldSimLastOH,simLastOHVar,lastOH,n);
  simlastOSVar:=NewVar(simLastOS,oldSimLastOS,simLastOSVar,lastOS,n);
  simTotalCostVar:=NewVar(simTotalCost,oldSimTotalCost,simTotalCostVar,totalCost,n);
  simInappVar:=NewVar(simInapp,oldSimInapp,simInappVar,inappAsset,n);
  simInappValVar:=NewVar(simInappVal,oldSimInappVal,simInappValVar,inappVal,n);
  simInitSSOHVar:=NewVar(simInitSSOH,oldSimInitSSOH,simInitSSOHVar,initSSOH,n);
  simInitSSOSVar:=NewVar(simInitSSOS,oldSimInitSSOS,simInitSSOSVar,initSSOS,n);
  simInitSSOrdersVar:=NewVar(simInitSSOrders,oldSimInitSSOrders,
                    simInitSSOrdersVar,initSSOrders,n);

end;

procedure DisplayQtrSimStats (var qtrACWTBOArry,qtrACWTArry,qtrSMAArry,
                    qtrInvestArry,qtrInappArry,cumACWTBOArry,
                    cumACWTArry,cumSMAArry:qtrStatArry;numberOfReps,
                    numberOfQtrs:integer;runDescript:descriptType;
                    silverSSType:char;var startSSQtr,endSSQtr:integer);
```

```
var t:integer;
   statOutFile:text;
   statFileName:string;

begin
  clrscr;
  write('Write Quarterly Statistics to a File? (Y or N): ');
  if Get_Answer then begin
      repeat
        writeln;
        write ('Enter Path and Filename: ');
        readln (statFileName);
        writeln;
        writeln ('Path and FileName entered: ',statFileName);
        writeln;
        write ('Change Path and FileName entered? (Y or N): ');
      until not(Get_Answer);
      assign(statOutFile,statFileName);
      rewrite (statOutFile);
      for t:=startSSQtr to endSSQtr do begin
        ConfInv(qtrACWTBOArry[t].Variance,qtrACWTBOArry[t].Mean,
              qtrACWTBOArry[t].CIHigh,qtrACWTBOArry[t].CILow,numberRep);
        ConfInv(qtrACWTArry[t].Variance,qtrACWTArry[t].Mean,
              qtrACWTArry[t].CIHigh,qtrACWTArry[t].CILow,numberRep);
        ConfInv(qtrSMAArry[t].Variance,qtrSMAArry[t].Mean,
              qtrSMAArry[t].CIHigh,qtrSMAArry[t].CILow,numberRep);
        ConfInv(qtrInvestArry[t].Variance,qtrInvestArry[t].Mean,
              qtrInvestArry[t].CIHigh,qtrInvestArry[t].CILow,numberRep);
        ConfInv(cumACWTBOArry[t].Variance,cumACWTBOArry[t].Mean,
              cumACWTBOArry[t].CIHigh,cumACWTBOArry[t].CILow,numberRep);
        ConfInv(cumACWTArry[t].Variance,cumACWTArry[t].Mean,
              cumACWTArry[t].CIHigh,cumACWTArry[t].CILow,numberRep);
        ConfInv(cumSMAArry[t].Variance,cumSMAArry[t].Mean,
              cumSMAArry[t].CIHigh,cumSMAArry[t].CILow,numberRep);
        ConfInv(qtrInappArry[t].Variance,qtrInappArry[t].Mean,
              qtrInappArry[t].CIHigh,qtrinappArry[t].CILow,numberRep);

      end;
      if silverSSType ='0' then begin
        writeln(statOutFile,' MOD SILVER (VARIABLE FORECASTS)');
      end else begin
        writeln(statOutFile,' MOD SILVER (FIXED FORECASTS)');
      end;
      writeln(statOutFile,' Description: ',runDescript);
      writeln(statOutFile);
      writeln(statOutFile,' QUARTERLY DATA:');
      writeln(statOutFile,' QTR  ACWTBO      CI       ACWT       CI ' );
      for t:=startSSQtr to endSSQtr do begin
        writeln(statOutFile,t:4,
                    qtrACWTBOArry[t].Mean:8:2,
```

```
                    qtrACWTBOArry[t].CILow:8:2,
                    qtrACWTBOArry[t].CIHigh:8:2,
                    qtrACWTArry[t].Mean:8:2,
                    qtrACWTArry[t].CILow:8:2,
                    qtrACWTArry[t].CIHigh:8:2);
      end;
      writeln(statOutFile);
      writeln(statOutFile,' QTR  SMA       CI     Invest      CI');
      for t:=startSSQtr to endSSQtr do begin
        writeln(statOutFile,t:4,
                    qtrSMAArry[t].Mean:8:2,
                    qtrSMAArry[t].CILow:8:2,
                    qtrSMAArry[t].CIHigh:8:2,
                    qtrInvestArry[t].Mean:8:2,
                    qtrInvestArry[t].CILow:8:2,
                    qtrInvestArry[t].CIHigh:8:2);
      end;
      writeln(statOutFile);
      writeln(statOutFile,' CUMULATIVE QUARTERLY DATA:');
      writeln(statOutFile,' QTR  ACWTBO      CI       ACWT       CI ' );
      for t:=startSSQtr to endSSQtr do begin
        writeln(statOutFile,t:4,
                    cumACWTBOArry[t].Mean:8:2,
                    cumACWTBOArry[t].CILow:8:2,
                    cumACWTBOArry[t].CIHigh:8:2,
                    cumACWTArry[t].Mean:8:2,
                    cumACWTArry[t].CILow:8:2,
                    cumACWTArry[t].CIHigh:8:2);
      end;
      writeln(statOutFile);
      writeln(statOutFile,' QTR  SMA       CI     Qtrly INAPP  CI');
      for t:=startSSQtr to endSSQtr do begin
        writeln(statOutFile,t:4,
                    cumSMAArry[t].Mean:8:2,
                    cumSMAArry[t].CILow:8:2,
                    cumSMAArry[t].CIHigh:8:2,
                    qtrInappArry[t].Mean:8:2,
                    qtrInappArry[t].CILow:8:2,
                    qtrInappArry[t].CIHigh:8:2);
      end;

      close(statOutFile);
    end;
end;

procedure DisplaySimStats (var simACWTBO, simACWT, simSMA, simInvest,
                    simOrderCount, simLastOH, simlastOS,simTotalCost,
                    simACWTBOVar, simACWTVar, simSMAVar,
                    simInvestVar,simOrderCountVar,
                    simLastOHVar,simlastOSVar,simTotalCostVar,
```

```
                    simInapp,simInappVar,simInappVal,
                    simInappValVar,simInitSSOH,simInitSSOS,
                    simInitSSOrders,simInitSSOHVar,simInitSSOSVar,
                    simInitSSOrdersVar:real;
                  var n:integer;
                  initInv,initOS,initOrders,initSSOH,
                  initSSOS,initSSOrders:integer;
                  outputType:char; hour1,minute1,second1,hdSec1,
                  hour2,minute2,second2,hdSec2:word);

var simACWTBOHi,simACWTHi,simSMAHi,simInvestHi,simOrderCountHi,
   simLastOHHi,simLastOSHi,simACWTBOLo,simACWTLo,simSMALo,
   simInvestLo,simOrderCountLo,simLastOHLo,simLastOSLo,
   simTotalCostHi,simTotalCostLo,simInappLo,simInappHi,
   simInappValLo,simInappValHi, simInitSSOHHi,simInitSSOHLo,
   simInitSSOSHi,simInitSSOSLo,simInitSSOrdersHi,
   simInitSSOrdersLo:real;


begin

  ConfInv(simACWTBOVar, simACWTBO, simACWTBOHi,simACWTBOLo,n);
  ConfInv(simACWTVar, simACWT, simACWTHi, simACWTLo,n);
  ConfInv(simSMAVar, simSMA, simSMAHi, simSMALo,n);
  ConfInv(simInvestVar, simInvest, simInvestHi, simInvestLo,n);
  ConfInv(simOrderCountVar, simOrderCount, simOrderCountHi,
       simOrderCountLo,n);
  ConfInv(simLastOHVar, simLastOH, simLastOHHi, simLastOHLo,n);
  ConfInv(simLastOSVar, simLastOS, simLastOSHi, simLastOSLo,n);
  ConfInv(simTotalCostVar, simTotalCost, simTotalCostHi, simTotalCostLo,n);
  ConfInv(simInappVar,simInapp, simInappHi, simInappLo,n);
  ConfInv(simInappValVar,simInappVal, simInappValHi, simInappValLo,n);
  ConfInv(simInitSSOHVar, simInitSSOH, simInitSSOHHi, simInitSSOHLo,n);
  ConfInv(simInitSSOSVar, simInitSSOS, simInitSSOSHi, simInitSSOSLo,n);
  ConfInv(simInitSSOrdersVar, simInitSSOrders, simInitSSOrdersHi,
       simInitSSOrdersLo,n);

  writeln
(outputfile,'**********************************************************************
************');
  writeln (outputfile,'Init OH Qty: ',initInv:8,
       '  Init SS OH Qty: ',simInitSSOH:8:2,'  (',simInitSSOHLo:0:2,
                              ',',simInitSSOHHi:0:2,')');
  writeln (outputfile,'Init OS Qty: ',initOS:8,
       '  Init SS OS Qty: ',simInitSSOS:8:2,'  (',simInitSSOSLo:0:2,
                             ',',simInitSSOSHi:0:2,')');
  writeln (outputfile,'Init Orders: ',initOrders:8,
       '  Init SS Orders: ',simInitSSOrders:8:2,'  (',simInitSSOrdersLo:0:2,
                               ',',simInitSSOrdersHi:0:2,')');
```

```
  writeln(outputfile);
  writeln(outputfile,'Simulation Final Statistics');
  writeln(outputfile);
  writeln(outputfile,'      ACWTBO  ACWT  SMA  Orders  INVEST  End OH  End OS');
  writeln(outputfile,'     ',simACWTBO:7:2,simACWT:7:2,simSMA:7:2,simOrderCount:8:2,
       simInvest:8:2,simLastOH:8:2,simLastOS:8:2);
  writeln(outputfile,'Low   ',simACWTBOLo:7:2,simACWTLo:7:2,simSMALo:7:2,
      simOrderCountLo:8:2,simInvestLo:8:2,simLastOHLo:8:2,
       simLastOSLo:8:2);
  writeln(outputfile,'High  ',simACWTBOHi:7:2,simACWTHi:7:2,simSMAHi:7:2,
       simOrderCountHi:8:2,simInvestHi:8:2,simLastOHHi:8:2,
       simLastOSHi:8:2);
  writeln(outputfile);
  writeln(outputfile,'     Total Cost  INAPP INAPP Value');
  writeln(outputfile,'     ',simTotalCost:10:2,simInapp:8:2,simInappVal:10:2);
  writeln(outputfile,'Low   ',simTotalCostLo:10:2,simInappLo:8:2,simInappValLo:10:2);
  writeln(outputfile,'High  ',simTotalCostHi:10:2,simInappHi:8:2,simInappValHi:10:2);

  if n<30 then begin
    writeln(outputfile);
    writeln(outputfile,'Caution! The confidence level is based on a normality assumption.');
    writeln(outputfile,'Your sample has only ',n:3,' values');
  end;
  writeln
(outputfile,'**********************************************************************
************');
  writeln(outputfile,'Sim Start Time   ',hour1,':',minute1,':',second1,':',hdSec1,
              '  Sim End Time    ',hour2,':',minute2,':',second2,':',hdSec2);
   if outputType ='1' then HitToCont;
end;



begin    {main}
  textcolor(14);
  stop:=FALSE;
  simCount:=0;
  Frontscreen;
  Runtype (distrType,outputType,wkDataType,qtrDataType,frcstDataType,
       repStatType,silverSSType,numberOfQtrs,numberOfWks,numberOfReps,

negBinS,seedIndex,startSSQtr,endSSQtr,meanDemand,varDemand,negBinP,inputfile,outputfi
le,
       frcst,mad,seeds,outFileName,runDescript);
  repeat
    rewrite (outputfile);
    simCount:=simCount+1;
    currSeed:=0;
    n:=0;
    GetTime( hour1,minute1,second1,hdSec1);
```

```
InitializeStatArrays(qtrACWTBOArry,qtrACWTArry,qtrSMAArry,
                qtrInvestArry,qtrInappArry,cumACWTBOArry,
                cumACWTArry,cumSMAArry);
for numberRep := 1 to numberOfReps do begin
  if seedType = '1' then begin
    if numberRep = 1 then begin
      for s:=1 to seedIndex do begin
        currSeed:=GetNextSeed(currSeed);
      end;
      SetSeed(currSeed);
    end else begin
      currSeed:=GetNextSeed (currSeed);
      SetSeed(currSeed);
    end;
  end else begin
    SetSeed(seeds[numberRep]);
  end;
  InitializeArrays (observ,meanDmdArry,varDmdArry,stepIndArry, trndIndArry,
                mkCodeArry,numberOfQtrs,numberOfWks,meanDemand,
                wklyObserv);
  LoadObserv (observ,frcst,mad,meanDmdArry,varDmdArry,wklyObserv,
          observType,distrType,numberOfQtrs,
          numberOfWks,numberRep,simCount,trendOn,stepOn,nmbrSteps,
          nmbrTrends, meanDemand,varDemand,inputfile,
          startstep, startrnd, endtrnd,stepmult, trendcoeff,
          trendpower);
  if numberRep = 1 then begin
    if simCount=1 then InitInput(unitPrice,PLT,P1,adminCost,obsol,
                  timePref,storage,shortCost,frcstErrCoeff,bufferMult,
                  PLTSigMuRatio,meanDemand,ROLowConst,
                  maxQtrs,minQtrs,maxDecl,initInv,initOS,initOrders);
    InputEdit(unitPrice,PLT,P1,adminCost,obsol,timePref,storage,shortCost,
          frcstErrCoeff,bufferMult,PLTSigMuRatio,meanDemand,
          ROLowConst,maxQtrs,minQtrs,maxDecl,initInv,initOS,initOrders);
  end;
  if numberRep=1 then PrintHeader(unitPrice,PLT,P1,adminCost,obsol,
                          timePref,storage,shortCost,
                          frcstErrCoeff,bufferMult,PLTSigMuRatio,
                          meanDemand,varDemand,negbinP,
                          ROLowConst,maxQtrs,minQtrs,maxDecl,negBinS,
                          seedIndex,initInv,initOS,initOrders,
                          outputfile,outputType,
                          distrType,silverSSType,
                          outFileName,runDescript,
                          nmbrSteps,nmbrTrends,startSSQtr,
                          endSSQtr,stepMult,
                          trendCoeff,trendPower,startStep,
                          starTrnd,endTrnd);
  Forecast (observ,frcst,mad, stepIndArry, trndIndArry,
        mkCodeArry,numberOfQtrs,numberRep,unitPrice);
```

```
      if frcstDataType='1' then DisplayFrcstOutput (observ, frcst, mad,
                                    stepIndArry, trndIndArry,
                                    mkCodeArry,numberOfQtrs,initInv,
                                    numberRep,outputType);
      SDR(OSHeap,BOHeap,wklyObserv,frcst,meanDmdArry,observ,mad,
         numberOfQtrs,initInv,initOS,initOrders,startSSQtr,endSSQtr,
         initSSOH,initSSOS,initSSOrders,meanDemand,TWUS,ACWTBO,
         ACWT,SMA,Invest,orderCount,lastOH,lastOS,totalCost,
         inappAsset,inappVal,wkDataType,
         qtrDataType,outputType,silverSSType,unitPrice,PLT,P1,adminCost,
         obsol,timePref,storage,shortCost,frcstErrCoeff,bufferMult,PLTSigMuRatio,

ROLowConst,maxQtrs,minQtrs,maxDecl,qtrACWTBOArry,qtrACWTArry,qtrSMAArry,
         qtrInvestArry,qtrInappArry,
         cumACWTBOArry,cumACWTArry,cumSMAArry,numberRep);
      if repStatType = '1' then DisplayRepStats (ACWTBO, ACWT, SMA, Invest,
orderCount,
                                    lastOH,lastOS,totalCost,
                                    inappAsset,inappVal,
                                    outputType );
      CalcSimStats(ACWTBO, ACWT, SMA, Invest,orderCount,lastOH,lastOS,
                totalCost,inappAsset,inappVal,initSSOH,initSSOS,
                 initSSOrders,n,simACWTBO,simACWT,simSMA,simInvest,
                simOrderCount,simLastOH,simLastOS,simTotalCost,
                simACWTBOVar,simACWTVar,simSMAVar,simInvestVar,
                simOrderCountVar,simLastOHVar,simLastOSVar,
                simTotalCostVar,simInapp,simInappVar,
                simInappVal,simInappValVar,simInitSSOH,simInitSSOS,
                simInitSSOrders,simInitSSOHVar,simInitSSOSVar,
                simInitSSOrdersVar);
    end; {for}
    GetTime(hour2,minute2,second2,hdSec2);
    DisplaySimStats( simACWTBO,simACWT,simSMA,simInvest,simOrderCount,
                simLastOH,simLastOS,simTotalCost,
                simACWTBOVar, simACWTVar,
                simSMAVar, simInvestVar,simOrderCountVar,
                simLastOHVar,simlastOSVar,simTotalCostVar,
                simInapp,simInappVar,simInappVal,simInappValVar,
                simInitSSOH,simInitSSOS,simInitSSOrders,simInitSSOHVar,
                simInitSSOSVar,simInitSSOrdersVar,
                n,initInv,initOS,initOrders,initSSOH,
                initSSOS,initSSOrders,outputType,hour1,minute1,
                second1,hdSec1,hour2,minute2,second2,hdSec2);
    close (outputfile);
    DisplayQtrSimStats (qtrACWTBOArry,qtrACWTArry,qtrSMAArry,
                  qtrInvestArry,qtrInappArry,
                  cumACWTBOArry,cumACWTArry,
                  cumSMAArry,numberOfReps,numberOfQtrs,
                  runDescript,silverSSType,startSSQtr,endSSQtr);
    RunAgain (outputfile,runDescript,outputType,
```

```
            frcst,mad,stop,outFileName);
  until stop;
  textcolor(15);
end. {main}
```

```
{$M $4000,0,0} {$r+} {$N+,E+} {$G+} {Q+}
program UICP_Simulator (input,ouput);

uses dos, crt, toolbox, unirand, PDUnit, pqueue;

type quarterArray=array [1..120] of real;
    weeklyArray=array [1..1560] of real;
    qtrIntArray=array [1..120] of integer;
    changeRealArry = array [1..10] of real;
    changeIntArry = array [1..10] of integer;
    pd82field=string[15];
    descriptType=string[40];
    statRecord = record
                   Mean:real;
                   Variance:real;
                   CIHigh:real;
                   CILow:real;
                 end;
    qtrStatArry=array [1..120] of statRecord;

const COEFF1=1.386;
     POWER1=0.746;
     COEFF2=3.869;
     POWER2=1.378;
     MAXPLT=14.0;
     MINPLT=2.0;
     ERROR=1.00000000000000E-0010;

var wklyObserv:weeklyArray;
   observ, frcst, mad, meanDmdArry, varDmdArry, EOQArry, ROLevelArry,
   APSRArry, attainRisk, SSADDBO, SSADD, SSSMA:quarterArray;
   stepIndArry, trndIndArry,mkCodeArry:qtrIntArray;
   cumACWTBOArry,cumACWTArry,cumSMAArry:qtrStatArry;
   qtrACWTBOArry,qtrACWTArry,qtrSMAArry,qtrInvestArry,qtrInappArry,
   qtrSSADDBOArry,qtrSSADDArry,qtrSSSMAArry:qtrStatArry;
   observType,distrType,outputType,seedtype,wkDataType,qtrDataType,
   PDDataType,RunPD86Type,repStatType:char;
   numberRep,i,n,s,numberOfReps,numberOfQtrs,numberOfWks,markCode,initInv,
   initOS,initOrders,simCount,seedIndex,negBinS,startSSQtr,endSSQtr,
   initSSOH,initSSOS,initSSOrders:integer;
   meanDemand, varDemand,PLTSigMuRatio,realval,negBinP,meanRisk:real;
   trendOn,StepOn,nmbrSteps, nmbrTrends:integer;
   TWUS:longint;
   inputfile,outputfile:text;
   stringval:pd82field;
   stop:boolean;
   startstep, startrnd, endtrnd: changeIntArry;
   stepmult, trendcoeff, trendpower: changeRealArry;
   hour1,minute1,second1,hdSec1,hour2,minute2,second2,hdSec2:word;
   outFileName:string;
```

```
    OSHeap,BOHeap:PriorityQueueType;
    ACWTBO,ACWT,SMA,Invest,orderCount,lastOH,lastOS,totalCost,
    inappAsset,inappVal:real;
    simACWTBO, simACWT, simSMA, simInvest, simOrderCount,simLastOH,
    simLastOS:real;
    simACWTBOVar,simACWTVar,simSMAVar,simInvestVar,simOrderCountVar,
    simLastOHVar,simLastOSVar,simTotalCost,simTotalCostVar,
    simInapp,simInappvar,simInappVal,simInappValVar,simMeanRisk,
    simMeanRiskVar,simInitSSOH,simInitSSOS,simInitSSOrders,
    simInitSSOHVar,simInitSSOSVar,simInitSSOrdersVar:real;
    runDescript:descriptType;
    currSeed:longint;


procedure Frontscreen;

begin
   clrscr;
   writeln;
   writeln;
   writeln;
   writeln;
   writeln;
   writeln;
   writeln ('                ********************************************');
   writeln ('                *   UICP LEVELS FORECASTING   *');
   writeln ('                *         SIMULATOR           *');
   writeln ('                *      FOR CONSUMABLES        *');
   writeln ('                *                             *');
   writeln ('                *   G. C. Robillard LT,SC     *');
   writeln ('                *                             *');
   writeln ('                *     Revised: 9/02/93         *');
   writeln ('                ********************************************');
   Delay(1500);  {For 1500 ms}
   clrscr;
end;

procedure runtype (var distrType,outputType,wkDataType,qtrDataType,
                      PDDataType,RunPD86Type,repStatType:char;
                   var numberOfQtrs,numberOfWks,numberOfReps,
                      negBinS,seedIndex,startSSQtr,endSSQtr:integer;
                   var meanDemand, varDemand,negBinP:real;
                   var inputfile,outputfile: text;
                   var frcst,mad: quarterArray;
                   var seeds:seedArryType;
                   var outFileName:string;
                   var runDescript:descriptType);
var done: boolean;
    i,maxStart:integer;
    demandInFile: string;
```

```
begin
  writeln;
  writeln ('  *** THIS SCREEN WILL ALLOW SELECTION OF RUN TYPE OPTIONS ***');
  done:=FALSE;
  writeln;
  writeln; writeln;
  write  ('Enter the number of replications (from 1 to 20000) to be run : ' );
  numberOfReps:=Get_Integer(1,20000);
  writeln;
  repeat
    writeln ('Random Number Generator Seed Selection: ');
    writeln;
    writeln ('  1 - Default Seeds (unique seed for each replication)');
    writeln ('  2 - Select Seeds (max number of replications is 100)');
    writeln;
    write ('Choice: ');
    seedtype:=readkey;
    writeln (seedtype);
    writeln;
    case seedtype of
      '1': begin
           done:=TRUE;
           maxStart:=20001-NumberOfReps;
           write('Enter Random Seed Start Index (1 to ',maxStart:2,'): ');
           seedIndex:=Get_Integer(1,maxStart);
           end;
      '2': begin
           done:=TRUE;
           if NumberOfReps > 100 then NumberOfReps:=100;
           for i := 1 to numberOfReps do begin
             write ('Enter Seed value for replication ',i,' : ');
             seeds[i]:=Get_LongInt(1,2147483646);
             writeln;
          end;  {for}
          end
    end
  until done=TRUE;
  clrscr;
  writeln ('         **** RUN SELECTION OPTIONS CONTINUED ****');
  writeln;
  writeln;
  write('Enter Run Description: ');
  readln (runDescript);
  writeln;
  write  ('Enter the number of simulation quarters (max 120): ' );
  numberOfQtrs:=Get_Integer(1,120);
  numberOfWks:=13*NumberOfQtrs;
  writeln;
  write  ('Enter the start of simulation SS (collect stats) quarter (max ',numberOfQtrs:3,'): '
);
```

```
  startSSQtr: = Get_Integer(1,numberOfQtrs);
  writeln;
  write  ('Enter the end of simulation SS (collect stats) quarter (max ',numberOfQtrs:3,'): '
);
  endSSQtr: = Get_Integer(startSSQtr,numberOfQtrs);
  writeln;
  done: = FALSE;
  repeat
    writeln ('Type of Distribution: ');
    writeln;
    writeln ('  1 - Normal');
    writeln ('  2 - Poisson');
    writeln ('  3 - Neg Binomial');
    writeln;
    write ('Choice: ');
    distrType: = readkey;
    writeln (distrType);
    writeln;
    case distrType of
      '1': begin
          done: = TRUE;
          write ('Enter quarterly mean demand: ');
          meanDemand: = Get_Real(0.0001,999999.0);
          writeln;
          write ('Enter demand variance: ');
          varDemand: = Get_Real(0.0001,999999.0);
          writeln
          end;
      '2': begin
          done: = TRUE;
          write ('Enter quarterly mean demand: ');
          meanDemand: = Get_Real(0.0001,999999.0);
          varDemand: = meanDemand;
          writeln;
          end;
      '3': begin
          done: = TRUE;
          repeat
            writeln;
            write ('Enter parameter p  (0 < p < 1): ');
            negBinP: = Get_Real(0.0001,0.9999);
            writeln;
            write ('Enter parameter s (s =  1,2,3 ... ) : ');
            negBinS: = Get_Integer(1,100);
            writeln;
            meanDemand: = (negbinS + (1-negBinP))/negBinP;
            varDemand: = (negBinS + (1-negBinP))/(sqr(negBinP));
            writeln('The quarterly mean is:     ',meanDemand:8:2);
            writeln('The demand variance is: ',varDemand:8:2);
            writeln;
```

```
          write('Change Initial Neg Binomial Parameters? (Y or N): ');
        until not(Get_Answer);
        end;
  end
until done=TRUE;
frcst[1]:=meanDemand;
mad[1]:=COEFF1*exp(POWER1*ln(frcst[1]));
done:=FALSE;
clrscr;
writeln ('        **** RUN SELECTION OPTIONS CONTINUED ****');
repeat
  writeln;
  writeln ('Send Output to: ');
  writeln;
  writeln ('  1 - Screen');
  writeln ('  2 - File');
  writeln;
  write ('Choice: ');
  outputType:=readkey;
  writeln (outputType);
  case outputType of
    '1': begin
          done:=TRUE;
          assign(outputfile,'con');
        end;
    '2': begin
          done:=TRUE;
          repeat
            writeln;
            write ('Enter Path and Filename: ');
            readln (outFileName);
            writeln;
            writeln ('Path and FileName entered: ',outFileName);
            writeln;
            write ('Change Path and FileName entered? (Y or N): ');
          until not(Get_Answer);
          assign(outputfile,outFileName);
        end;
    end;
until done=TRUE;
clrscr;
writeln ('        **** RUN SELECTION OPTIONS CONTINUED ****');
wkDataType:='0';
writeln;
write('Include Weekly SDR Data? (Y or N): ');
if Get_Answer then  wkDataType:='1';
qtrDataType:='0';
writeln;
write('Include Quarterly SDR Data? (Y or N): ');
if Get_Answer then  qtrDataType:='1';
```

```
  PDDataType:='0';
  writeln;
  write('Include Quarterly demand, forecast and PD82/86 Data? (Y or N): ');
  if Get_Answer then  PDDataType:='1';
  RunPD86Type:='0';
  writeln;
  write('Run PD86 Steady State Projections ? (Y or N): ');
  if Get_Answer then  RunPD86Type:='1';
  repStatType:='0';
  writeln;
  write('Include Replication Statistics? (Y or N): ');
  if Get_Answer then  repStatType:='1';
end;


procedure RunAgain (var outputfile:text;var runDescript:descriptType;
                    outputType:char;
                    var frcst,mad:quarterArray;
                    var stop:boolean;
                    var outFileName:string);

var demandInFile: string;
   done1:boolean;
begin
  stop:=FALSE;
  clrscr;
  writeln ('          **** RE-RUN SIMULATION OPTIONS SCREEN ****');
  writeln;
  writeln('Re-running the simulation will maintain the same run-type parameters, but will');
  writeln('allow the user to change the destination (output) file and vary NIIN');
  writeln('and model parameters.');
  writeln;
  write('Do you wish to re-run the simulation? (Y or N): ');
  if Get_Answer then begin
    writeln;
    write('Change Run Description? (Y or N): ');
      if Get_Answer then begin
         writeln;
         write ('Enter Run Description: ');
         readln (runDescript);
      end;
    if outputType='2' then begin
      writeln;
      write('Change Output File? (Y or N): ');
      if Get_Answer then begin
        repeat
          writeln;
          write ('Enter Output Path and Filename: ');
          readln (outFileName);
          writeln;
```

```
            writeln ('Path and FileName entered: ',outFileName);
            writeln;
            write ('Change Path and FileName entered? (Y or N): ');
          until not(Get_Answer);
          assign(outputfile,outFileName);
        end;
      end;
    end else begin
        stop:=TRUE;
    end;
    clrscr;
end;


function GetMarkCode (t,oldMark:integer; frcst, unitPrice:real):integer;

begin
  if t=1 then begin
    if frcst < 0.25 then getMarkCode:=0;
    if (frcst >= 0.25) and (frcst < 2.0) then begin
      if (unitPrice >= 300.00) then begin
        getMarkCode:=3;
      end else begin
        getMarkCode:=1;
      end;
    end;
    if frcst >= 2.0 then begin
      if (unitPrice*frcst) >= 600.0 then begin
        getMarkCode:=4;
      end else begin
        getMarkCode:=2
      end;
    end;
  end else begin
    getMarkCode:=oldMark;
    if oldMark = 0 then begin
      if frcst >= 0.5 then begin
        if (unitPrice >= 300.00) then begin
          getMarkCode:=3;
        end else begin
          getMarkCode:=1;
        end;
      end;
      if frcst >=3 then begin
        if (unitPrice*frcst) >= 600.0 then begin
          getMarkCode:=4;
        end else begin
          getMarkCode:=2
        end;
      end;
```

```
    end;
    if (oldMark=1) or (oldMark=3) then begin
      if frcst >=3 then begin
        if (unitPrice*frcst) >= 600.0 then begin
          getMarkCode:=4;
        end else begin
          getMarkCode:=2
        end;
      end else if unitPrice <= 200 then begin
          getMarkCode:=1;
      end else if unitPrice >= 400 then begin
          getMarkCode:=3;
      end;
      if frcst <= 0.25 then getMarkCode:=0;
    end;
    if (oldMark=2) or (oldMark=4) then begin
      if frcst <= 1.0 then begin
        if (unitPrice >= 300.00) then begin
          getMarkCode:=3;
        end else begin
          getMarkCode:=1;
        end;
      end else if (unitPrice*frcst) >= 800.00 then begin
          getMarkCode:=4;
      end else if (unitPrice*frcst) <= 400.00 then begin
         getMarkCode:=2;
      end;
      if frcst <= 0.25 then getMarkCode:=0;
    end;
  end
end;


procedure InitializeStatArrays(var qtrACWTBOArry,qtrACWTArry,qtrSMAArry,
                        qtrInvestArry,qtrInappArry,cumACWTBOArry,
                        cumACWTArry,cumSMAArry:qtrStatArry);

var t:integer;

begin
  for t:=1 to numberOfQtrs do begin
    qtrACWTBOArry[t].Mean:=0.0; qtrACWTBOArry[t].Variance:=0.0;
    qtrACWTBOArry[t].CIHigh:=0.0; qtrACWTBOArry[t].CILow:=0.0;
    qtrACWTArry[t].Mean:=0.0; qtrACWTArry[t].Variance:=0.0;
    qtrACWTArry[t].CIHigh:=0.0; qtrACWTArry[t].CILow:=0.0;
    qtrSMAArry[t].Mean:=0.0; qtrSMAArry[t].Variance:=0.0;
    qtrSMAArry[t].CIHigh:=0.0; qtrSMAArry[t].CILow:=0.0;
    qtrINVESTArry[t].Mean:=0.0; qtrINVESTArry[t].Variance:=0.0;
    qtrINVESTArry[t].CIHigh:=0.0; qtrINVESTArry[t].CILow:=0.0;
    qtrInappArry[t].Mean:=0.0; qtrInappArry[t].Variance:=0.0;
```

```
    qtrInappArry[t].CIHigh:=0.0; qtrInappArry[t].CILow:=0.0;
    cumACWTBOArry[t].Mean:=0.0; cumACWTBOArry[t].Variance:=0.0;
    cumACWTBOArry[t].CIHigh:=0.0; cumACWTBOArry[t].CILow:=0.0;
    cumACWTArry[t].Mean:=0.0; cumACWTArry[t].Variance:=0.0;
    cumACWTArry[t].CIHigh:=0.0; cumACWTArry[t].CILow:=0.0;
    cumSMAArry[t].Mean:=0.0; cumSMAArry[t].Variance:=0.0;
    cumSMAArry[t].CIHigh:=0.0; cumSMAArry[t].CILow:=0.0;
  end;
end;


procedure InitializeArrays (var observ,meanDmdArry,varDmdArry, EOQArry,
                    ROLevel, APSRArry, attainRisk, SSADDBO,
                    SSADD, SSSMA:quarterArray;
                    var  stepIndArry, trndIndArry,mkCodeArry: qtrIntArray;
                    numberOfQtrs,numberOfWks:integer;
                    meanDemand:real;
                    var wklyObserv:weeklyArray);
var t:integer;

begin
  for t:=1 to numberOfQtrs do begin
    observ[t]:=0.0;
    meanDmdArry[t]:=0.0;
    varDmdArry[t]:=0.0;
    EOQArry[t]:=0.0;
    ROLevel[t]:=0.0;
    APSRArry[t]:=0.0;
    attainRisk[t]:=0.0;
    SSADDBO[t]:=0.0;
    SSADD[t]:=0.0;
    SSSMA[t]:=0.0;
    stepIndArry[t]:=0;
    trndIndArry[t]:=0;
    mkCodeArry[t]:=0;
  end;
  for t:=1 to (numberOfWks) do begin
    wklyObserv[t]:=0.0;
  end;
end;


procedure LoadObserv (var observ,frcst,mad,meanDmdArry,varDmdArry:quarterArray;
                var wklyObserv:weeklyArray;
                observType,distrType:char;
                numberOfQtrs,numberOfWks,repNum,simCount:integer;
                var trendInd,stepInd,nmbrSteps, nmbrTrends:integer;
                meanDemand, varDemand:real;
                var inputfile:text;
                var startstep, startrnd, endtrnd: changeIntArry;
```

```
                var stepmult, trendcoeff, trendpower: changeRealArry);

var SS:char;
   i, t, min, startQtr, endQtr,observWeek,s:integer;
   randnorm, currMeanDmd, initTrendMean, coeffVar,qtrCum,
   wkObserv,qtrObserv,p:real;
   demandInFile:string;

begin
  if (repNum = 1) and (simCount = 1) then begin
    for i:=1 to 10 do begin
      startstep[i]:=0; startrnd[i]:=0; endtrnd[i]:=0;
      stepmult[i]:=0.0; trendcoeff[i]:=0.0; trendpower[i]:=0.0;
    end;
    nmbrSteps:=0;
    nmbrTrends:=0;
  end; {if}
  currMeanDmd:=meanDemand;
  coeffVar:=sqrt(varDemand)/meanDemand;
  for t:=0 to (numberOfQtrs) do begin
    if (t=0) and (repNum = 1) and (simCount=1) then begin
      SS:='Y';
      writeln;
      write('Do you wish to vary mean demand rate over time? (Y or N): ');
      if Get_Answer then begin
        SS:='N';
        stepInd:=0;
        trendInd:=0;
        clrscr;
        writeln;
        writeln ('          *** Mean Demand Variants ***   ');
        writeln;
        writeln ('You have the option to vary mean demand rate over time.  If the normal');
        writeln ('distribution was selected, variance will also change to maintain your');
        writeln ('original variance to mean ratio.  You may choose between step change');
        writeln ('or trend or any combination of the events.  If more than one event is');
        writeln ('chosen to occur at the same time, step changes will occur first.');
        writeln ('A maximum of 10 occurances of each event is allowed.  Time of');
        writeln ('variation is specified by quarter.');
        writeln ;

        SS:='Y';
        write ('Do you still wish to vary mean demand rate over time? (Y or N): ');
        if Get_Answer then begin
          SS:='N';
          clrscr;
          writeln('          *** Step Changes Screen ***');
          writeln;
          write ('Do you wish to have step increases or decreases? (Y or N): ');
```

```
if Get_Answer then stepInd:=1;
if stepInd=1 then begin
  writeln;
  write('Enter the number of steps changes desired (max 10): ');
  nmbrSteps:=Get_Integer(1,10);
  writeln;
  writeln('The step function is of the form:  Mean(t) = A * Mean(t-1).');
  writeln('You must specify the value of "A" for each step.');
  min:=1;
  for i:=1 to nmbrSteps do begin
    writeln;
    writeln ('Step ',i,':');
    writeln;
    write ('Step Qtr: ');
    startQtr:=Get_Integer(min,numberOfQtrs);
    startstep[i]:=startQtr;
    writeln;
    write ('Step Multiplier (A): ');
    stepmult[i]:=Get_Real(0.00001,9999.0);
    writeln;
    min:=startQtr;
  end;
end;
clrscr;
writeln('              *** Trend Setting Screen ***');
writeln;
write ('Do you wish to have trends? (Y or N):');
if Get_Answer then trendInd:=1;
if trendInd=1 then begin
  writeln;
  write('Enter the number of trend periods desired (max 10): ');
  nmbrtrends:=Get_Integer(1,10);
  writeln;
  writeln('The trend function is of the form:');
  writeln('              Mean(t) = InitTrendMean * ( 1 + A * t(0) ** B)');
  writeln('where t(0) is reset to "1" at the beginning of each trend period');
  writeln('and InitTrendMean is the Mean at the beginning of the trend period.');
  writeln('Parameters A and B must be specified for each trend period.');
  min:=1;
  for i:=1 to nmbrtrends do begin
    writeln;
    writeln ('Trend ',i,':');
    writeln;
    write ('Start Qtr: ');
    startQtr:=Get_Integer(min,numberOfQtrs);
    startrnd[i]:=startQtr;
    writeln;
    write ('End Qtr: ');
    endQtr:=Get_Integer(startQtr,numberOfQtrs);
    endtrnd[i]:=endQtr;
```

```
            writeln;
            write ('Trend coefficent (A): ');
            trendcoeff[i]:=Get_Real(-9999.0,9999.0);
            writeln;
            write ('Trend power (B): ');
            trendpower[i]:=Get_Real(-9999.0,9999.0);
            writeln;
            min:=endQtr+1;
          end;
        end;
      end;
    end;
  end else if t > 0 then begin
    if SS='Y' then begin
      meanDmdArry[t]:=meanDemand;
      if (distrType='1') or (distrType='3') then begin
        varDmdArry[t]:=varDemand;
      end else begin
        varDmdArry[t]:=currMeanDmd;
      end;
    end else begin
      if stepInd = 1 then begin
        for i:=1 to nmbrSteps do begin
          if t = startstep[i] then currMeanDmd:=stepmult[i]*currMeanDmd;
        end;
      end;
      if trendInd = 1 then begin
        for i:=1 to nmbrTrends do begin
          if t = startrnd[i] then initTrendMean:=currMeanDmd;
          if (t >= startrnd[i]) and (t <= endtrnd[i]) then begin
            currMeanDmd:=initTrendMean*(1+trendcoeff[i]*
                    (exp(trendpower[i]*ln(t-startrnd[i]+1))));
            if currMeanDmd < 0.0 then currMeanDmd:=0.0;
          end;
        end;
      end;
      meanDmdArry[t]:=currMeanDmd;
      if (distrType='1') or (distrType='3') then begin
        varDmdArry[t]:=sqr(coeffVar*currMeanDmd);
      end else begin
        varDmdArry[t]:=currMeanDmd;
      end;
    end;
    if distrType='1' then begin
      randnorm:=GetNormal;
      qtrObserv:=round(meanDmdArry[t]+(randnorm*sqrt(varDmdArry[t])));
      if qtrObserv < 0.0 then qtrObserv:=0.0;
      for i:=1 to round(qtrObserv) do begin
        observWeek:=GetUniformInt(13);
        wklyObserv[(t-1)*13+observWeek]:=
```

```
                wklyObserv[(t-1)*13+observWeek]+1;
          end;
        end else if distrType='2' then begin
          qtrObserv:=GetPoisson(meanDmdArry[t]);
          for i:=1 to round(qtrObserv) do begin
            observWeek:=GetUniformInt(13);
            wklyObserv[(t-1)*13+observWeek]:=
                wklyObserv[(t-1)*13+observWeek]+1;
          end;
        end else if distrType='3' then begin
          p:=(meanDmdArry[t])/(varDmdArry[t]);
          s:=round((sqr(meanDmdarry[t]))/(varDmdArry[t]-meanDmdArry[t]));
          if (p>ERROR) and (p<(1-Error)) then begin
            qtrObserv:=GetNegBin(p,s);
          end else begin
            qtrObserv:=0.0;
          end;
          for i:=1 to round(qtrObserv) do begin
            observWeek:=GetUniformInt(13);
            wklyObserv[(t-1)*13+observWeek]:=
                wklyObserv[(t-1)*13+observWeek]+1;
          end;
        end;
        observ[t]:=qtrObserv;
      end; {else,if}
    end; {for}
    clrscr;
end;

procedure Forecast (var observ, frcst, mad:quarterArray;
                    var stepIndArry, trndIndArry,mkCodeArry: qtrIntArray;
                    numberOfQtrs,repNum:integer; unitPrice:real);

const ALPHA=0.1;
      STEPBOUND1=3.0;
      STEPBOUND2=2.0;

var upper, lower, sum, sampleMean, sampleStdDev, stdDevToMean:real;
    upInd, downInd, stepInd, trendInd, trendUp,
    trendDn, t, i, j, W, S, table:integer;
    kendTest, lowDemand:boolean;

begin
  writeln('Running Replication # ',repNum);
  mkCodeArry[1]:=getMarkCode (1,0,frcst[1],unitPrice);
  upInd:=0;downInd:=0;
  for t:=2 to numberOfQtrs do begin          {Compute quarterly forecast}
    lowDemand:=FALSE;
    trendInd:=0;
    stepInd:=0;
```

```
    if ((mkCodeArry[t-1] = 0) or (mkCodeArry[t-1] = 1) or (mkCodeArry[t-1] =3)) then
lowDemand:=TRUE;
    if lowDemand then begin
      upper:=STEPBOUND1*frcst[t-1];
      lower:=0.0;
    end else begin
      upper:=frcst[t-1]+1.25*mad[t-1]*STEPBOUND2;
      lower:=frcst[t-1]-1.25*mad[t-1]*STEPBOUND2;
    end;
    if (lowDemand and (observ[t-1] < 5)) or
      ((observ[t-1] < upper) and (observ[t-1] >= lower)) then begin
      upInd:=0;
      downInd:=0;
      frcst[t]:=ALPHA*observ[t-1]+(1-ALPHA)*frcst[t-1];
      mad[t]:=ALPHA*(abs(observ[t-1]-frcst[t-1]))+(1-ALPHA)*mad[t-1];
    end else begin
      if ((observ[t-1] > upper) and (upInd=1)) or
        ((observ[t-1] < lower) and (downInd=1)) then begin
        if t>4 then begin
          frcst[t]:=(observ[t-4]+observ[t-3]+observ[t-2]+observ[t-1])/4;
        end else if t = 4 then begin
          frcst[t]:=(observ[t-3]+observ[t-2]+observ[t-1])/3;
        end else if t = 3 then begin
           frcst[t]:=(observ[t-2]+observ[t-1])/2;
        end;
        if frcst[t] > ERROR then begin
          mad[t]:=COEFF1*exp(POWER1*ln(frcst[t]));
        end else begin
          mad[t]:=0.0;
        end;
        stepInd:=1;
        upInd:=0;
        downInd:=0;
      end else begin
        if ((observ[t-1] > upper) and (upInd=0)) then begin
          upInd:=1;
          frcst[t]:=frcst[t-1];
          mad[t]:=mad[t-1];
        end else begin
          if ((observ[t-1] < lower) and (downInd=0)) then begin
            downInd:=1;
            frcst[t]:=frcst[t-1];
            mad[t]:=mad[t-1];
          end;
        end;
      end;
    end;
    if (t>4) and (stepInd=0) then begin     {Conduct Kendall Trend Test}
      sum:=0.0;
      if t <= 8 then begin
```

```
  for i:=1 to t-1 do begin
    sum:=sum+observ[i];
  end;
  sampleMean:=sum/(t-1);
  sum:=0.0;
  for i:=1 to t-1 do begin
    sum:=sum+sqr(observ[i]-sampleMean);
  end;
  sampleStdDev:=sqrt(sum/(t-2));
end else begin
  for i:=t-8 to t-1 do begin
    sum:=sum+observ[i];
  end;
  sampleMean:=sum/8;
  sum:=0.0;
  for i:=t-8 to t-1 do begin
    sum:=sum+sqr(observ[i]-sampleMean);
  end;
  sampleStdDev:=sqrt(sum/7);
end;
if sampleMean > 0.0 then begin
  stdDevToMean:=sampleStdDev/sampleMean
end else begin
  stdDevToMean:=99999.0
end;
kendTest:=false;
if (sampleMean >= 3.0) and (stdDevToMean <= 1.75) then begin
  kendTest:=true;
  if stdDevToMean > 1.0 then begin
    table:=3;
  end else begin
    table:=2;
  end;
end;
if ((sampleMean >= 1.0) and (sampleMean < 3.0)) and
  (stdDevToMean <= 1.75) then begin
  kendTest:=true;
  if stdDevToMean > 1.25 then begin
    table:=3;
  end else begin
    table:=2;
  end;
end;
if ((sampleMean >= 0.125) and (sampleMean < 1.0)) and
  (stdDevToMean <= 2.00) then begin
  kendTest:=true;
  table:=2;
end;
if kendTest=true then begin       {Conduct Kendall S-Test for Trend}
  W:=8;
```

```
if (sampleMean >= 3.0) and (sampleMean < 9.0) then begin
  if (stdDevToMean < 0.30) then W:=6;
  end;
if (sampleMean >= 9.0) and (sampleMean < 20.0) then begin
  if (stdDevToMean < 0.93) then W:=6;
  if (stdDevToMean < 0.28) then W:=4;
  end;
if (sampleMean >= 20.0) then begin
  if (stdDevToMean < 0.53) then W:=6;
  if (stdDevToMean < 0.28) then W:=4;
  end;
if W > (t-1) then W:=((t-1) div 2)*2;
S:=0;
for i:=(t-W) to (t-2) do begin      {Compute Kendall S-Statistic}
  for j:=(i+1) to (t-1) do begin
    if observ[i] < observ[j] then S:=S+1;
    if observ[i] > observ[j] then S:=S-1;
  end;
end; {for}
if table = 2 then begin
  if W = 4 then begin
    trendUp:=4; trendDn:=-4;
  end;
  if W = 6 then begin
    trendUp:=9; trendDn:=-9;
  end;
  if W = 8 then begin
    trendUp:=13; trendDn:=-13;
  end;
end else begin
  if W = 4 then begin
    trendUp:=6; trendDn:=-6;
  end;
  if W = 6 then begin
    trendUp:=11; trendDn:=-11;
  end;
  if W = 8 then begin
    trendUp:=16; trendDn:=-16;
  end;
end; {if}
trendInd:=0;
if S >= trendUp then trendInd:=1;
if S <= trendDn then trendInd:=1;
if trendInd =1 then begin
  sum:=0.0;
  for i:=(t-4) to (t-1) do begin
    sum:=sum+observ[i];
    end;
  frcst[t]:=sum/4;
  if frcst[t] > ERROR then begin
```

```
            mad[t]:=COEFF1*exp(POWER1*ln(frcst[t]));
          end else begin
            mad[t]:=0.0;
          end;
        end; {if}
      end; {if}
    end; {if}
  mkCodeArry[t]:=getMarkCode (t,mkCodeArry[t-1],frcst[t],unitPrice);
  stepIndArry[t]:=stepInd;
  trndIndArry[t]:=trendInd;
  end; {for}
end;

procedure LoadLevels (var frcst, mad, observ, EOQArry, ROLevelArry,
                APSRArry, attainRisk, SSADDBO, SSADD, SSSMA:quarterArray;
                var qtrSSADDBOArry,qtrSSADDArry,
                qtrSSSMAArry:qtrStatArry;
                var mkCodeArray:qtrIntArray;
                var numberOfQtrs:integer;
                prbBrkPt,numberRep:integer; meanDemand,PLTSigMuRatio:real;
                var meanRisk:real;
                PDDataType,RunPD86Type:char);

var A023B,BRLDC,B011A,B019A,B023C,B023D,B073,M,PPV,APSR,B014A,
   B019,B021,BRLDCU: real;

   PD82str1: string[24];
   PD82str2, PD82str3, PD82str4, PD82str5, PD82str6, PD82str7,
   PD82str8: string[255];

   PD86str1: string[24];
   PD86str2, PD86str3, PD86str4, PD86str5, PD86str6, PD86str7,
   PD86str8: string[255];
   PD86str9: string[60];

   infile,outfile:text;
   LTVar,oldQtrSSADDBO,oldQtrSSADD,oldQtrSSSMA:real;
   t:integer;

begin
  meanRisk:=0.0;
  for t:=1 to numberOfQtrs do begin
    gotoXY(1,3);
    write('Quarter # ',t);
    assign (infile,'c:\tp\pd82in.fil');
    reset (infile);
    read(infile,PD82str1, PD82str2, PD82str3, PD82str4, PD82str5, PD82str6,
        PD82str7, PD82str8);
    close (infile);
    B023D:=frcst[t];                    {current quarterly forecast}
```

```
A023B:=meanDemand;
if t>4 then begin
      A023B:=(observ[t-4]+observ[t-3]+observ[t-2]+observ[t-1])/4;
    end else if t = 4 then begin
      A023B:=(observ[t-3]+observ[t-2]+observ[t-1])/3;
    end else if t = 3 then begin
      A023B:=(observ[t-2]+observ[t-1])/2;
    end;
if A023B <= 0.0 then A023B:=1.0;
strTemp:=copy(PD82str2,46,15);  B011A:=StringToReal(StrTemp);
B023C:=B011A*B023D;
PPV:=B023C;
delete (PD82str2,1,15);
insert (NumToString(A023B),PD82str2,1);
delete (PD82str2,121,15);
insert (NumToString(B023D),PD82str2,121);
delete (PD82str2,106,15);
insert (NumToString(B023C),PD82str2,106);
delete (PD82str5,91,15);
insert (NumToString(PPV),PD82str5,91);
M:=mkCodeArry[t];                              {current mark code}
delete (PD82str4,241,15);
insert (NumToString(M),PD82str4,241);
if (mkCodeArry[t] = 2) or (mkCodeArry[t]=4) then begin
  LTVar:=sqr(PLTSigMuRatio*B011A);   {default = 1.57*B011A}
  B019A:=B011A*(sqr(mad[t])*1.57)+(sqr(frcst[t]))*LTVar;
end else begin
  if abs(B023C)<ERROR then B023C:=0.0;
  if B023C=0.0 then begin
    B019A:=0.0
  end else begin
    B019A:=COEFF2*exp(POWER2*ln(B023C))
  end;
end;
delete (PD82str2,76,15);
insert (NumToString(B019A),PD82str2,76);
if mkCodeArry[t] = 0 then begin
  BRLDC:=3;
end else begin
  if prbBrkPt = 0 then begin
    BRLDC:=5;
  end else begin
    if B023C < prbBrkPt then begin
      BRLDC:=4;
    end else begin
      BRLDC:=5;
    end;
  end;
end;
delete (PD82str2,16,15);
```

```
insert (NumToString(BRLDC),PD82str2,16);
assign (outfile,'c:\tp\pd82in.fil');
rewrite (outfile);
writeln(outfile,PD82str1, PD82str2, PD82str3, PD82str4, PD82str5, PD82str6,
      PD82str7, PD82str8);
close (outfile);
SwapVectors;
exec ('c:\tp\PPD82KR0.exe','c:\tp pd82in.fil pd82out.fil ' );
SwapVectors;
if DosError <> 0 then  begin
   writeln;
   Sound(220);
   delay (300);
   NoSound;
   writeln ('Dos error #', DosError);
   HitToCont;
end;
assign (infile,'c:\tp\pd82out.fil');
reset (infile);
read(infile,PD82str1, PD82str2, PD82str3, PD82str4, PD82str5, PD82str6,
    PD82str7, PD82str8);
close (infile);
strTemp:=copy(PD82str7,196,15); B019:=StringToReal(StrTemp);
ROLevelArry[t]:=B019;
strTemp:=copy(PD82str7,226,15); B021:=StringToReal(StrTemp);
EOQArry[t]:=B021;
strTemp:=copy(PD82str7,121,15); BRLDCU:=StringToReal(StrTemp);
strTemp:=copy(PD82str7,61,15); APSR:=StringToReal(StrTemp);
APSRArry[t]:=APSR;
strTemp:=copy(PD82str7,181,15); B014A:=StringToReal(StrTemp);
attainRisk[t]:=B014A;
meanRisk:=meanRisk + B014A;

if (PDDataType = '1') or (RunPD86Type = '1') then begin
  InitPD86File;
  SwapVectors;
  exec ('c:\tp\PPD86KR4.exe','c:\tp pd86in.fil pd86out.fil ' );
  SwapVectors;
  if DosError <> 0 then  begin
    writeln;
    Sound(220);
    delay (300);
    NoSound;
    writeln ('Dos error #', DosError);
    HitToCont;
  end;
  assign (infile,'c:\tp\pd86out.fil');
  reset (infile);
  read(infile,PD86str1, PD86str2, PD86str3, PD86str4, PD86str5, PD86str6,
      PD86str7, PD86str8, PD86str9);
```

```
        close (infile);
        strTemp:=copy(PD86str8,166,15); SSADDBO[t]:=StringToReal(StrTemp);
        strTemp:=copy(PD86str8,181,15); SSADD[t]:=StringToReal(StrTemp);
        strTemp:=copy(PD86str8,196,15); SSSMA[t] :=StringToReal(StrTemp);

        oldQtrSSADDBO:=qtrSSADDBOArry[t].Mean;

qtrSSADDBOArry[t].Mean:=NewMean(qtrSSADDBOArry[t].Mean,SSADDBO[t],numberRep);
        qtrSSADDBOArry[t].Variance:=NewVar(qtrSSADDBOArry[t].Mean,oldQtrSSADDBO,
                        qtrSSADDBOArry[t].Variance,SSADDBO[t],numberRep);
        oldQtrSSADD:=qtrSSADDArry[t].Mean;
        qtrSSADDArry[t].Mean:=NewMean(qtrSSADDArry[t].Mean,SSADD[t],numberRep);
        qtrSSADDArry[t].Variance:=NewVar(qtrSSADDArry[t].Mean,oldQtrSSADD,
                        qtrSSADDArry[t].Variance,SSADD[t],numberRep);
        oldQtrSSSMA:=qtrSSSMAArry[t].Mean;
        qtrSSSMAArry[t].Mean:=NewMean(qtrSSSMAArry[t].Mean,SSSMA[t],numberRep);
        qtrSSSMAArry[t].Variance:=NewVar(qtrSSSMAArry[t].Mean,oldQtrSSSMA,
                        qtrSSSMAArry[t].Variance,SSSMA[t],numberRep);


    end;
  end;
  meanRisk:=meanRisk/numberOfQtrs;
end;


procedure SDR(var OSHeap,BOHeap:PriorityQueueType;
          var wklyObserv:weeklyArray;
          var EOQArry,ROLevelArry,observ:quarterArray;
          var numberOfQtrs,initInv,initOS,initOrders,startSSQtr,endSSQtr,
            initSSOH,initSSOS,initSSOrders:integer;
          var PLT,meanDemand,PLTSigMuRatio,
            obsol,timePref,storage,shortCost,adminCost:real;
          var TWUS:longint;
          var ACWTBO,ACWT,SMA,Invest,orderCount,lastOH,lastOS,totalCost,
            inappAsset,inappVal:real;
          wkDataType,qtrDataType,outputType:char;
          var qtrACWTBOArry,qtrACWTArry,qtrSMA,qtrInvestArry,qtrInappArry,
            cumACWTBOArry,cumACWTArry,cumSMAArry:qtrStatArry;
            numberRep:integer);

var  wklyBO,wklyOS:datarecord;
    amtBO,amtRecv,receipt,wklyDemand,date,BOqtr,endQtr:integer;
    i,t,wk,qtr,sizeOS,sizeBO:integer;
    randnorm,randPLT,wklyInvest,qtrInvest,repInvest,
    holdCost,cumSSHoldCost,qtrInapp,twoYearAmt,intLength,startInt,SSOrderCount:real;
    flag1,flag2:boolean;
    BOFill,dmdTot,SSOSTot,OSCurr,BOTot,BOCurr,OHcurr,IPcurr:integer;
    oldCumACWTBO,oldCumACWT,oldCumSMA,oldQtrInvest,oldQtrACWTBO,oldQtrInapp,
    oldQtrACWT,oldQtrSMA,ACWTBOvalue,ACWTvalue,SMAvalue:real;
```

```
    qtrTWUSArry,qtrBOTotArry,qtrBOFillArry:qtrIntArray;

begin
  holdCost:=unitPrice*(obsol+timePref+storage)/52;
  InitializePriorityQueue(OSHeap);InitializePriorityQueue(BOHeap);
  initInv:=round(EOQArry[1]/2 + ROLevelArry[1]-(meanDemand * PLT));
  if initInv <1 then initInv:=1;
  OHCurr:=initInv;
  initOS:=round(PLT * meanDemand);
  if initOS < 1 then initOS:=1;
  initOrders:=round(initOS/EOQArry[1]);
  initOS:=initOrders*round(EOQArry[1]);
  OSCurr:=initOS;
  intLength:= (13*PLT)/initOrders;
  startInt:=0.0;
  for i:=1 to initOrders do begin
    wklyOS.Qty:=round(EOQArry[1]);
    wklyOS.Week:=round((startInt + (i*intLength))/2);
    InsertPriorityQueue(OSHeap,wklyOS);
    startInt:=startInt+intLength;
  end;
  IPCurr:=OHCurr + OSCurr;

  if (qtrDataType = '1') or (wkDataType = '1') then begin
    writeln(outputfile);
    writeln(outputfile,'SDR Data        Initial OH Inv:= ',initInv);
    writeln(outputfile,'-------------------------------------------------------------------');
  end;

  for t:=1 to numberOfQtrs do begin
    qtrTWUSArry[t]:=0;
    qtrBOTotArry[t]:=0;
    qtrBOFillArry[t]:=0;
  end;

  BOCurr:=0;
  replnvest:= 0.0;
  date:=1;

  for qtr:= 1 to numberOfQtrs do begin

    if (qtr >= startSSQtr) and (qtr <= endSSQtr) then begin
      if wkDataType = '1' then begin
        writeln(outputfile);
        writeln(outputfile,'QTR  WK   REC   DEM    BO   OS    OH   IP  ORDCNT');
      end;
     end;

    if qtr=startSSQtr then begin
      initSSOH:=OHCurr;
```

```
      initSSOS:=OSCurr;
      initSSOrders:=SizePriorityQueue(OSHeap);
      cumSSHoldCost:=0.0;
      SSOSTot:=0;
      SSOrderCount:=0.0;
    end;
    qtrInvest:=0.0;
    wklyInvest:= 0.0;

    for wk:= 1 to 13 do begin
      wklyDemand:=round(wklyObserv[date]);
      receipt:=0;
      amtRecv:=0;
      amtBO:=0;
      wklyBO.Qty:=0;
      wklyBO.Week:=date;
      wklyOS.Qty:=0;
      flag1:=FALSE; flag2:=FALSE;

      if not (EmptyPriorityQueue(OSHeap)) then begin          {receive}
        repeat
          if CurrWeek(OSHeap) = date then begin
            amtRecv:=ExtractQty(OSHeap);
            receipt:=amtRecv;
            OSCurr:= OSCurr - amtRecv;
            while (amtRecv > 0) and not (EmptyPriorityQueue(BOHeap)) do begin
              if CurrQty(BOHeap) <= amtRecv then begin
                amtBO:=CurrQty(BOHeap);
                amtRecv:= amtRecv - amtBO;
                BOCurr:= BOCurr - amtBO;
                if (CurrWeek(BOHeap) mod 13) = 0 then begin
                  BOqtr:=(CurrWeek(BOHeap) div 13);
                end else begin
                  BOqtr:=(CurrWeek(BOHeap) div 13)+1;
                end;
                qtrBOFillArry[BOqtr]:=qtrBOFillArry[BOqtr]+amtBO;
                qtrTWUSArry[BOqtr]:=qtrTWUSArry[BOqtr]+(amtBo*(date -
ExtractWeek(BOHeap)));
              end else begin
                BOHeap.HeapArray[1].Qty:= BOHeap.HeapArray[1].Qty - amtRecv;
                if (BOHeap.HeapArray[1].Week mod 13) = 0 then begin
                  BOqtr:=(BOHeap.HeapArray[1].Week) div 13;
                end else begin
                  BOqtr:=((BOHeap.HeapArray[1].Week) div 13)+1;
                end;
                qtrTWUSArry[BOqtr]:=qtrTWUSArry[BOqtr]+(amtRecv*(date -
BOHeap.HeapArray[1].Week));
                BOCurr:= BOCurr - amtRecv;
                qtrBOFillArry[BOqtr]:=qtrBOFillArry[BOqtr]+amtRecv;
                amtRecv:= 0;
```

```
        end; {if}
      end; {while}
      OHCurr:=OHCurr + amtRecv;
    end;
    if EmptyPriorityQueue(OSHeap) then flag2:= TRUE
    else if currWeek(OSHeap) <> date then flag1:=TRUE;
  until flag1 or flag2;
end; {if  receive}

if wklyDemand > 0 then begin              {issue}
  if wklyDemand > OHCurr then begin
    wklyBO.Qty:= wklyDemand - OHCurr;
    OHCurr:=0;
    InsertPriorityQueue(BOHeap,wklyBO);
    qtrBOTotArry[qtr]:=qtrBOTotArry[qtr]+wklyBO.Qty;
    BOCurr:=BOCurr + wklyBO.Qty;
  end else begin
    OHCurr:= OHCurr - wklyDemand;
  end;
end; {if  issue}


IPCurr:= OHCurr + OSCurr - BOCurr;        {order}
if IPCurr <= ROLevelArry[qtr] then begin
  wklyOS.Qty:=round(ROLevelArry[qtr] + EOQArry[qtr]) + BOCurr -
          (OHCurr + OSCurr);
  randnorm:=GetNormal;
  randPLT:=abs(PLT+(randnorm*PLTSigMuRatio*PLT));
  if randPLT > MAXPLT then begin
    randPLT:=MAXPLT;
  end else if randPLT < MINPLT then begin
    randPLT:=MINPLT
  end;
  wklyOS.Week:=date + round(randPLT*13) + 1;
  InsertPriorityQueue(OSHeap,wklyOS);
  OSCurr:= OSCurr + wklyOS.Qty;
  if (qtr >= startSSQtr) and (qtr <= endSSQtr) then begin
    SSOrderCount:=SSOrderCount + 1.0;
    SSOSTot:= SSOSTot + wklyOS.Qty;
  end;
end; {if}

if (qtr >= startSSQtr) and (qtr <= endSSQtr) then begin
  if wkDataType = '1' then begin
    writeln(outputfile,qtr:3,date:5,receipt:6,wklyDemand:6,BOCurr:6,
          OSCurr:6,OHCurr:6,IPCurr:6,SSOrderCount:6:0);
    if (outputType = '1') and ((wk mod 13) = 0) then HitToCont;
  end;
end;
```

```
  receipt:=0;
  date:=date+1;
  wklyInvest:= wklyInvest + OSCurr + OHCurr;
  if (qtr >= startSSQtr) and (qtr <= endSSQtr) then begin
    cumSSHoldCost:=cumSSHoldCost+OHCurr*holdCost;
  end;
end; {for week}

qtrInvest:= wklyInvest/13;
if (qtr >= startSSQtr) and (qtr <= endSSQtr) then begin
   repInvest:= repInvest + qtrInvest;
end;

oldQtrInvest:=qtrInvestArry[qtr].Mean;
qtrInvestArry[qtr].Mean:=NewMean(qtrInvestArry[qtr].Mean,qtrInvest,numberRep);
qtrInvestArry[qtr].Variance:=NewVar(qtrInvestArry[qtr].Mean,oldQtrInvest,
                        qtrInvestArry[qtr].Variance,qtrInvest,numberRep);

twoYearAmt:=0.0;
if qtr<numberOfQtrs then begin
    twoYearAmt:=8*frcst[qtr+1];
end;
qtrInapp:=OHCurr-twoYearAmt;
if qtrInapp < 0.0 then qtrInapp:=0.0;
oldQtrInapp:=qtrInappArry[qtr].Mean;
qtrInappArry[qtr].Mean:=NewMean(qtrInappArry[qtr].Mean,qtrInapp,numberRep);
qtrInappArry[qtr].Variance:=NewVar(qtrInappArry[qtr].Mean,oldQtrInapp,
                        qtrInappArry[qtr].Variance,qtrInapp,numberRep);

if qtr = endSSQtr then begin
  invest:= repInvest/(endSSQtr-startSSQtr+1);
  lastOH:=OHCurr;
  lastOS:=OSCurr;
  inappAsset:=qtrInapp;
  inappVal:=qtrInapp*unitPrice;
  orderCount:=SSOrderCount;
end;

if (qtr >= startSSQtr) and (qtr <= endSSQtr) then begin
  if (wkDataType='1') then begin
     writeln(outputfile);
     writeln(outputfile,'QTR   DMD    OH    IP    OS    BO   INVEST');
  end else  if qtrDataType = '1' then begin
    if (qtr=1) or (((qtr-1) mod 20 ) = 0)then begin
      writeln(outputfile);
      writeln(outputfile,'QTR   DMD    OH    IP    OS    BO  INVEST ');
    end;
  end;
  if qtrDataType = '1' then
     writeln(outputfile,qtr:3,observ[qtr]:6:0,OHCurr:6,IPCurr:6,
```

```
          OSCurr:6,BOCurr:6,qtrInvest:8:2);
       if (outputType = '1') and (qtrDataType ='1') and (((qtr-1) mod 20) = 0) then
HitToCont;
    end;

  end; {for qtr}

  if not (EmptyPriorityQueue(OSHeap)) then begin    {adjust final qtr TWUS}
     while not (EmptyPriorityQueue(BOHeap)) do begin
        amtBO:=CurrQty(BOHeap);
        if (CurrWeek(BOHeap) mod 13) = 0 then begin
           BOqtr:=(CurrWeek(BOHeap) div 13);
        end else begin
           BOqtr:=(CurrWeek(BOHeap) div 13 + 1);
        end;
        qtrBOFillArry[BOqtr]:=qtrBOFillArry[BOqtr]+amtBO;
        qtrTWUSArry[BOqtr]:=qtrTWUSArry[BOqtr]+
              (amtBo*(OSHeap.HeapArray[1].Week - ExtractWeek(BOHeap)));
     end; {while}
  end;

  for t:= 1 to numberOfQtrs do begin
    if qtrBOFillArry[t] > 0 then begin
      oldQtrACWTBO:=qtrACWTBOArry[t].Mean;
      ACWTBOvalue:=(7*(qtrTWUSArry[t]/qtrBOFillArry[t]));
      qtrACWTBOArry[t].Mean:=NewMean(qtrACWTBOArry[t].Mean,
                                ACWTBOvalue,numberRep);
      qtrACWTBOArry[t].Variance:=NewVar(qtrACWTBOArry[t].Mean,oldQtrACWTBO,
                           qtrACWTBOArry[t].Variance,ACWTBOvalue,numberRep);
    end; {if}
    oldQtrACWT:=qtrACWTArry[t].Mean;
    if observ[t] > 0 then begin
      ACWTvalue:=(7*(qtrTWUSArry[t]/observ[t]));
    end else begin
      ACWTvalue:=0.0;
    end;
    qtrACWTArry[t].Mean:=NewMean(qtrACWTArry[t].Mean,
                           ACWTvalue,numberRep);
    qtrACWTArry[t].Variance:=NewVar(qtrACWTArry[t].Mean,oldQtrACWT,
                         qtrACWTArry[t].Variance,ACWTvalue,numberRep);
    oldQtrSMA:=qtrSMAArry[t].Mean;
    if observ[t] > 0 then begin
      SMAvalue:= (1-(qtrBOTotArry[t]/observ[t]));
    end else begin
      SMAvalue:=1.0;
    end;
    qtrSMAArry[t].Mean:=NewMean(qtrSMAArry[t].Mean,
                           SMAvalue,numberRep);
    qtrSMAArry[t].Variance:=NewVar(qtrSMAArry[t].Mean,oldQtrSMA,
                         qtrSMAArry[t].Variance,SMAvalue,numberRep);
```

```
    end; {for}

    dmdTot:=0;
    TWUS:=0;
    BOTot:=0;
    BOFill:=0;

    for qtr:=startSSQtr to endSSQtr do begin

      dmdTot:=dmdTot + round(observ[qtr]);
      TWUS:=TWUS + qtrTWUSArry[qtr];
      BOTot:=BOTot + qtrBOTotArry[qtr];
      BOFill:= BOFILL + qtrBOFillArry[qtr];

      if BOFill <> 0 then begin
        ACWTBO:=7*(TWUS/BOFill);
      end else begin
        ACWTBO:=0.0;
      end; {if}
      if dmdTot <> 0 then begin
        ACWT:=7*(TWUS/dmdTot);
        SMA:=1 - BOTot/dmdTot;
      end else begin
        ACWT:=0.0;
        SMA:=1.0;
      end; {if}

      oldCumACWTBO:=cumACWTBOArry[qtr].Mean;

cumACWTBOArry[qtr].Mean:=NewMean(cumACWTBOArry[qtr].Mean,ACWTBO,numberRep
);

cumACWTBOArry[qtr].Variance:=NewVar(cumACWTBOArry[qtr].Mean,oldCumACWTBO,
                          cumACWTBOArry[qtr].Variance,ACWTBO,numberRep);
      oldCumACWT:=cumACWTArry[qtr].Mean;
      cumACWTArry[qtr].Mean:=NewMean(cumACWTArry[qtr].Mean,ACWT,numberRep);
      cumACWTArry[qtr].Variance:=NewVar(cumACWTArry[qtr].Mean,oldCumACWT,
                          cumACWTArry[qtr].Variance,ACWT,numberRep);
      oldCumSMA:=cumSMAArry[qtr].Mean;
      cumSMAArry[qtr].Mean:=NewMean(cumSMAArry[qtr].Mean,SMA,numberRep);
      cumSMAArry[qtr].Variance:=NewVar(cumSMAArry[qtr].Mean,oldCumSMA,
                          cumSMAArry[qtr].Variance,SMA,numberRep);
    end;  {for}

    totalCost:=(initSSOH+initSSOS+SSOSTot)*unitPrice+SSOrderCount*adminCost+
          cumSSHoldCost+(TWUS/52*shortCost);

end;  {sdr}
```

```
procedure PrintHeader(prbBrkPt,negBinS,seedIndex:integer;meanDemand, varDemand,
                PLTSigMuRatio,negBinP:real;
                var outputfile:text;outputType,distrType:char;
                outFileName:string;runDescript:descriptType;
                nmbrSteps,nmbrTrends,startSSqtr,endSSQtr:integer;stepMult,trendCoeff,
                trendPower:changeRealArry;startStep,starTrnd,
                endTrnd:changeIntArry);

var i:integer;
   distrUsed:string[7];
   infile:text;
   Year,Month,Day,Dayofweek:word;
   C028 : string[1];

   A023B,B010,B011A,B020,B023C,B023D,B055,B057,B058,B061,B073,C008C,D025E,
   MSLQD,SCR,TD,TSDRS,V015R,V022,V101A,V102,V1034,V295: real;

   PD82str1: string[24];
   PD82str2, PD82str3, PD82str4, PD82str5, PD82str6, PD82str7,
   PD82str8: string[255];


begin
  distrUsed:=' Normal';
  if distrType = '2' then distrUsed:='Poisson';
  if distrType = '3' then distrUsed:='Neg Binomial';
  if outputType = '2' then begin
    writeln(outputfile,'                *** ',outFileName,' ***');
    writeln(outputfile);
    GetDate(Year,Month,Day,Dayofweek);
    writeln(outputfile,' Date: ',Month,'-',Day,'-',Year,'     Model: UICP - EOQ ');
  end;
  writeln(outputfile);
  writeln(outputfile,' Description: ',runDescript);
  writeln(outputfile);
  writeln(outputfile,' Initial simulation settings  ');
  writeln(outputfile);
  writeln(outputfile,'  Random number generator seed type:          ',seedtype);
  if seedType ='1' then begin
    writeln(outputfile,'  Random number seed start index:          ',seedIndex:6);
  end;
  writeln(outputfile,'  Type of demand distribution:            ', distrUsed);
  if distrType = '3' then begin
    writeln(outputfile,'  Neg Binomial Parameters:              p = ',negBinP:6:2);
    writeln(outputfile,'                                    s = ',negBinS:6);
  end;
  writeln(outputfile,'  Mean Demand:                          ',meanDemand:6:2);
  writeln(outputfile,'  Var Demand:                          ',varDemand:6:2);
  writeln(outputfile,'  Number of quarters to simulate:          ',numberOfQtrs:5 );
```

```
writeln(outputfile,'  Start Sim Steady State quarter:              ',startSSQtr:5 );
writeln(outputfile,'  End Sim Steady State quarter:                ',endSSQtr:5 );
writeln(outputfile,'  Number of replications of simulation to run: ',numberOfReps:5);
writeln(outputfile,'  Number of steps:                             ',nmbrSteps:5);
if nmbrSteps >0 then begin
  for i:=1 to nmbrSteps do begin
    writeln(outputfile,'    Step: ',i:2,' Step Qtr: ',startStep[i]:4,
          ' Mult: ',stepMult[i]:7:4);
  end;
end; {if}
writeln(outputfile,'  Number of trends:                            ',nmbrTrends:5);
if nmbrTrends >0 then begin
  for i:=1 to nmbrTrends do begin
     writeln(outputfile,'    Trend:',i:2,' Start Qtr: ',starTrnd[i]:3,
          ' Stop Qtr: ',endTrnd[i]:3,
          ' Coeff: ',trendCoeff[i]:7:4,' Power: ',trendPower[i]:7:4);
  end;
end; {if}
writeln(outputfile);
if outputType = '1' then begin
  HitToCont;
  clrscr;
end;
writeln(outputfile,' Initial parameter settings  ');
assign (infile,'c:\tp\pd82in.fil');
reset (infile);
read(infile,PD82str1, PD82str2, PD82str3, PD82str4, PD82str5, PD82str6,
   PD82str7, PD82str8);
close (infile);
C028:=copy(PD82str1,5,1);

strTemp:=copy(PD82str2,46,15);  B011A:=StringToReal(StrTemp);
strTemp:=copy(PD82str2,91,15);  B020:=StringToReal(StrTemp);
strTemp:=copy(PD82str2,121,15); B023D:=StringToReal(StrTemp);
strTemp:=copy(PD82str2,181,15); B055:=StringToReal(StrTemp);
strTemp:=copy(PD82str2,211,15); B057:=StringToReal(StrTemp);
strTemp:=copy(PD82str2,226,15); B058:=StringToReal(StrTemp);
strTemp:=copy(PD82str3,1,15);   B061:=StringToReal(StrTemp);
strTemp:=copy(PD82str3,31,15);  B073:=StringToReal(StrTemp);
strTemp:=copy(PD82str3,76,15);  C008C:=StringToReal(StrTemp);
strTemp:=copy(PD82str3,121,15); D025E:=StringToReal(StrTemp);
strTemp:=copy(PD82str5,31,15);  MSLQD:=StringToReal(StrTemp);
strTemp:=copy(PD82str5,181,15); SCR:=StringToReal(StrTemp);
strTemp:=copy(PD82str5,211,15); TD:=StringToReal(StrTemp);
strTemp:=copy(PD82str5,226,15); TSDRS:=StringToReal(StrTemp);
strTemp:=copy(PD82str5,241,15); V015R:=StringToReal(StrTemp);
strTemp:=copy(PD82str6,16,15);  V022:=StringToReal(StrTemp);
strTemp:=copy(PD82str6,106,15); V101A:=StringToReal(StrTemp);
strTemp:=copy(PD82str6,121,15); V102:=StringToReal(StrTemp);
strTemp:=copy(PD82str6,136,15); V1034:=StringToReal(StrTemp);
```

```
  strTemp:=copy(PD82str6,166,15); V295:=StringToReal(StrTemp);

  writeln (outputfile,'  Prob Break: ',PrbBrkPt:8, '     Min Risk  : ',V022:8:2);
  writeln (outputfile,'  Shelf Life:      ',C028,'     Max Risk  : ',V102:8:2);
  writeln (outputfile,'  Reqn Size : ',B073:8:0, '     Ord Cost  : ',V015R:8:2);
  writeln (outputfile,'  Unit Price: ',B055:8:2,  '     MSLQD     : ',MSLQD:8:2);
  writeln (outputfile,'  Procur LT : ',B011A:8:2,  '     Proc Meth : ',D025E:8:0);
  writeln (outputfile,'  PLT Sig/Mu: ',PLTSigMuratio:8:2, '     Shortage  : ',V1034:8:2);
  writeln (outputfile,'  Essential : ',C008C:8:2, '     R/O Low   : ',B020:8:2);
  writeln (outputfile,'  Mfg Set-Up: ',B058:8:2,  '     R/O Constr: ',V295:8:2);
  writeln (outputfile,'  Obsol Rate: ',B057:8:2,  '     Stor Rate : ',SCR:8:2);
  writeln (outputfile,'  Disc Rate : ',B061:8:2,  '     Time Pref : ',V101A:8:2);
  writeln (outputfile,'  Time SDRS : ',TSDRS:8:2, '     Today DT  : ',TD:8:0);

  if outputType = '1' then begin
    HitToCont;
    clrscr;
  end;
end;   {printheader}



procedure DisplayPDOutput (var  observ, frcst, mad, EOQArry, ROLevelArry,
                    APSRArry, attainRisk, SSADDBO, SSADD, SSSMA:quarterArray;
                    var  stepIndArry, trndIndArry,mkCodeArry:qtrIntArray;
                    numberOfQtrs,initInv,repNum:integer;
                    outputType:char);

var t:integer;

begin
  writeln (outputfile);
  writeln(outputfile,'Replication Number ',repNum);
  writeln(outputfile);
  writeln(outputfile,'PD82/86 Data');
  writeln(outputfile,'------------------------------------------------------------------------');
  for t:=1 to numberOfQtrs do begin
    if (t=1) or (((t-1) mod 20) = 0)then begin
      if (outputType='1') and (t>1) then HitToCont;
      writeln(outputfile);
      writeln (outputfile,'QTR   OBS  FRCST    MAD    Q  R/O   ADDBO    ADD   SMA
MK ST TR  APSR AttR');
    end;
    writeln (outputfile,t:3,observ[t]:6:0,frcst[t]:8:2,mad[t]:8:2,
          EOQArry[t]:6:0,ROLevelArry[t]:6:0,
          SSADDBO[t]:8:2,SSADD[t]:8:2,SSSMA[t]:6:2,mkCodeArry[t]:3,
          stepIndArry[t]:3,trndIndArry[t]:3,APSRArry[t]:6:2,
          attainRisk[t]:5:2);
  end;
  writeln (outputfile);
```

```
  if outputType= '1' then HitToCont;
end;

procedure DisplayRepStats (var ACWTBO, ACWT, SMA, Invest,orderCount,lastOH,
                    lastOS,totalCost,inappAsset,inappVal:real;
                    outputType:char );
begin
  if (numberRep = 1) or (outputType = '1') then begin
    writeln(outputfile);
    writeln
(outputfile,'**********************************************************************
****');
    writeln(outputfile,' Rep# ACWTBO   ACWT    SMA  INVEST  End OH    Tot Cost
Inapp');
  end;
  writeln(outputfile,numberRep:5,ACWTBO:7:2,ACWT:7:2,SMA:7:2,Invest:8:2,lastOH:8:0,'
 ',totalCost:10:2,inappAsset:8:0);
  if outputType ='1' then begin
    delay(1500);
    clrscr;
  end;
end;


procedure CalcSimStats(ACWTBO, ACWT, SMA, Invest,orderCount,lastOH,lastOS,
                totalCost,inappAsset,inappVal,meanRisk,initSSOH,
                initSSOS,initSSOrders:real;
                var n:integer;
                var simACWTBO,simACWT,simSMA,simInvest,
                   simOrderCount,simLastOH,simLastOS,
                   simTotalCost,simACWTBOVar,simACWTVar,
                   simSMAVar,simInvestVar,simOrderCountVar,
                   simLastOHVar,simLastOSVar,
                   simTotalCostVar,simInapp,
                   simInappVar,simInappVal,simInappValVar,
                   simMeanRisk,simMeanRiskVar,simInitSSOH,
                   simInitSSOS,simInitSSOrders,
                   simInitSSOHVar,simInitSSOSVar,
                   simInitSSOrdersVar:real);

var oldSimACWTBO,oldSimACWT,oldSimSMA,oldSimInvest,oldSimOrderCount,
  oldSimLastOH,oldSimLastOS,oldSimTotalCost,oldSimInapp,oldSimInappVal,
  oldSimMeanRisk,oldSimInitSSOH,oldSimInitSSOS,oldSimInitSSOrders:real;

begin
  if n=0 then begin
    simACWTBO:=0.0;simACWT:=0.0;simSMA:=0.0;simInvest:=0.0;
    simOrderCount:=0.0;simLastOH:=0.0;simLastOS:=0.0;simTotalCost:=0;
    simInapp:=0.0;simInappVal:=0.0;
    simACWTBOVar:=0.0;simACWTVar:=0.0;simSMAVar:=0.0;
```

```
    simInvestVar: = 0.0;simOrderCountVar: = 0.0;simLastOHVar: = 0.0;
    simLastOSVar: = 0.0;simTotalCostVar: = 0.0;
    simInappVar: = 0.0;simInappValVar: = 0.0;
    simMeanRisk: = 0.0;simMeanRiskVar: = 0.0;
    simInitSSOH: = 0.0;simInitSSOS: = 0.0;simInitSSOrders: = 0.0;
    simInitSSOHVar: = 0.0;simInitSSOSVar: = 0.0;simInitSSOrdersVar: = 0.0;
  end;

  n: = n + 1;
  oldSimACWTBO: = simACWTBO;oldSimACWT: = simACWT;oldSimSMA: = simSMA;
  oldSimInvest: = simInvest;oldSimOrderCount: = simOrderCount;
  oldSimLastOH: = simLastOH;oldSimLastOS: = simLastOS;
  oldSimTotalCost: = simTotalCost;oldSimInapp: = simInapp;
  oldSimInappVal: = simInappVal;
  oldSimMeanRisk: = simMeanRisk;
  oldSimInitSSOH: = simInitSSOH;
  oldSimInitSSOS: = simInitSSOS;
  oldSimInitSSOrders: = simInitSSOrders;

  simACWTBO: = NewMean(simACWTBO,ACWTBO,n);
  simACWT: = NewMean(simACWT,ACWT,n);
  simSMA: = NewMean(simSMA,SMA,n);
  simInvest: = NewMean(simInvest,Invest,n);
  simOrderCount: = NewMean(simOrderCount,orderCount,n);
  simLastOH: = NewMean(simLastOH,lastOH,n);
  simLastOS: = NewMean(simLastOS,lastOS,n);
  simTotalCost: = NewMean(simTotalCost,totalCost,n);
  simInapp: = NewMean(simInapp,inappAsset,n);
  simInappVal: = NewMean(simInappVal,inappVal,n);
  simMeanRisk: = NewMean(simMeanRisk,meanRisk,n);
  simInitSSOH: = NewMean(simInitSSOH,initSSOH,n);
  simInitSSOS: = NewMean(simInitSSOS,initSSOS,n);
  simInitSSOrders: = NewMean(simInitSSOrders,initSSOrders,n);

  simACWTBOVar: = NewVar(simACWTBO,oldSimACWTBO,simACWTBOVar,ACWTBO,n);
  simACWTVar: = NewVar(simACWT,oldSimACWT,simACWTVar,ACWT,n);
  simSMAVar: = NewVar(simSMA,oldSimSMA,simSMAVar,SMA,n);
  simInvestVar: = NewVar(simInvest,oldSimInvest,simInvestVar,Invest,n);

simOrderCountVar: = NewVar(simOrderCount,oldSimOrderCount,simOrderCountVar,orderCou
nt,n);
  simlastOHVar: = NewVar(simLastOH,oldSimLastOH,simLastOHVar,lastOH,n);
  simlastOSVar: = NewVar(simLastOS,oldSimLastOS,simLastOSVar,lastOS,n);
  simTotalCostVar: = NewVar(simTotalCost,oldSimTotalCost,simTotalCostVar,totalCost,n);
  simInappVar: = NewVar(simInapp,oldSimInapp,simInappVar,inappAsset,n);
  simInappValVar: = NewVar(simInappVal,oldSimInappVal,simInappValVar,inappVal,n);
  simMeanRiskVar: = NewVar(simMeanRisk,oldSimMeanRisk,simMeanRiskVar,meanRisk,n);
  simInitSSOHVar: = NewVar(simInitSSOH,oldSimInitSSOH,simInitSSOHVar,initSSOH,n);
  simInitSSOSVar: = NewVar(simInitSSOS,oldSimInitSSOS,simInitSSOSVar,initSSOS,n);
  simInitSSOrdersVar: = NewVar(simInitSSOrders,oldSimInitSSOrders,
```

```
                    simInitSSOrdersVar,initSSOrders,n);

end;

procedure DisplayQtrSimStats (var qtrACWTBOArry,qtrACWTArry,qtrSMAArry,
                    qtrInvestArry,qtrInappArry,cumACWTBOArry,
                    cumACWTArry,cumSMAArry,
                    qtrSSADDBOArry,qtrSSADDArry,
                    qtrSSSMAArry:qtrStatArry;numberOfReps,
                    numberOfQtrs:integer;runDescript:descriptType;
                    RunPD86Type:char;var startSSQtr,endSSQtr:integer);


var t:integer;
   statOutFile:text;
   statFileName:string;

begin
  clrscr;
  write('Write Quarterly Statistics to a File? (Y or N): ');
  if Get_Answer then begin
      repeat
        writeln;
        write ('Enter Path and Filename: ');
        readln (statFileName);
        writeln;
        writeln ('Path and FileName entered: ',statFileName);
        writeln;
        write ('Change Path and FileName entered? (Y or N): ');
      until not(Get_Answer);
      assign(statOutFile,statFileName);
      rewrite (statOutFile);
      for t:=startSSQtr to endSSQtr do begin
        ConfInv(qtrACWTBOArry[t].Variance,qtrACWTBOArry[t].Mean,
              qtrACWTBOArry[t].CIHigh,qtrACWTBOArry[t].CILow,numberRep);
        ConfInv(qtrACWTArry[t].Variance,qtrACWTArry[t].Mean,
              qtrACWTArry[t].CIHigh,qtrACWTArry[t].CILow,numberRep);
        ConfInv(qtrSMAArry[t].Variance,qtrSMAArry[t].Mean,
              qtrSMAArry[t].CIHigh,qtrSMAArry[t].CILow,numberRep);
        ConfInv(qtrInvestArry[t].Variance,qtrInvestArry[t].Mean,
              qtrInvestArry[t].CIHigh,qtrInvestArry[t].CILow,numberRep);
        ConfInv(cumACWTBOArry[t].Variance,cumACWTBOArry[t].Mean,
              cumACWTBOArry[t].CIHigh,cumACWTBOArry[t].CILow,numberRep);
        ConfInv(cumACWTArry[t].Variance,cumACWTArry[t].Mean,
              cumACWTArry[t].CIHigh,cumACWTArry[t].CILow,numberRep);
        ConfInv(cumSMAArry[t].Variance,cumSMAArry[t].Mean,
              cumSMAArry[t].CIHigh,cumSMAArry[t].CILow,numberRep);
        ConfInv(qtrInappArry[t].Variance,qtrInappArry[t].Mean,
```

```
        qtrInappArry[t].CIHigh,qtrinappArry[t].CILow,numberRep);

end;
writeln(statOutFile,' UICP (EOQ) MODEL');
writeln(statOutFile,' Description: ',runDescript);
writeln(statOutFile);
writeln(statOutFile,' QUARTERLY DATA:');
writeln(statOutFile,' QTR  ACWTBO      CI       ACWT       CI ' );
for t:=startSSQtr to endSSQtr do begin
  writeln(statOutFile,t:4,
                qtrACWTBOArry[t].Mean:8:2,
                qtrACWTBOArry[t].CILow:8:2,
                qtrACWTBOArry[t].CIHigh:8:2,
                qtrACWTArry[t].Mean:8:2,
                qtrACWTArry[t].CILow:8:2,
                qtrACWTArry[t].CIHigh:8:2);
end;
writeln(statOutFile);
writeln(statOutFile,' QTR  SMA       CI      Invest      CI');
for t:=startSSQtr to endSSQtr do begin
  writeln(statOutFile,t:4,
                qtrSMAArry[t].Mean:8:2,
                qtrSMAArry[t].CILow:8:2,
                qtrSMAArry[t].CIHigh:8:2,
                qtrInvestArry[t].Mean:8:2,
                qtrInvestArry[t].CILow:8:2,
                qtrInvestArry[t].CIHigh:8:2);
end;
writeln(statOutFile);
writeln(statOutFile,' CUMULATIVE QUARTERLY DATA:');
writeln(statOutFile,' QTR  ACWTBO      CI       ACWT       CI ' );
for t:=startSSQtr to endSSQtr do begin
  writeln(statOutFile,t:4,
                cumACWTBOArry[t].Mean:8:2,
                cumACWTBOArry[t].CILow:8:2,
                cumACWTBOArry[t].CIHigh:8:2,
                cumACWTArry[t].Mean:8:2,
                cumACWTArry[t].CILow:8:2,
                cumACWTArry[t].CIHigh:8:2);
end;
writeln(statOutFile);
writeln(statOutFile,' QTR  SMA       CI      Qtrly INAPP  CI');
for t:=startSSQtr to endSSQtr do begin
  writeln(statOutFile,t:4,
                cumSMAArry[t].Mean:8:2,
                cumSMAArry[t].CILow:8:2,
                cumSMAArry[t].CIHigh:8:2,
                qtrInappArry[t].Mean:8:2,
                qtrInappArry[t].CILow:8:2,
                qtrInappArry[t].CIHigh:8:2);
```

```
        end;

        if RunPD86Type = '1' then begin
          for t:=startSSQtr to endSSQtr do begin
            ConfInv(qtrSSADDBOArry[t].Variance,qtrSSADDBOArry[t].Mean,
                 qtrSSADDBOArry[t].CIHigh,qtrSSADDBOArry[t].CILow,numberRep);
            ConfInv(qtrSSADDArry[t].Variance,qtrSSADDArry[t].Mean,
                 qtrSSADDArry[t].CIHigh,qtrSSADDArry[t].CILow,numberRep);
            ConfInv(qtrSSSMAArry[t].Variance,qtrSSSMAArry[t].Mean,
                 qtrSSSMAArry[t].CIHigh,qtrSSSMAArry[t].CILow,numberRep);
          end;
          writeln(statOutFile);
          writeln(statOutFile,' QTR SSADDBO      CI      SSADD      CI ' );
          for t:=startSSQtr to endSSQtr do begin
            writeln(statOutFile,t:4,
                         qtrSSADDBOArry[t].Mean:8:2,
                         qtrSSADDBOArry[t].CILow:8:2,
                         qtrSSADDBOArry[t].CIHigh:8:2,
                         qtrSSADDArry[t].Mean:8:2,
                         qtrSSADDArry[t].CILow:8:2,
                         qtrSSADDArry[t].CIHigh:8:2);
          end;
          writeln(statOutFile);
          writeln(statOutFile,' QTR  SSSMA      CI  ');
          for t:=startSSQtr to endSSQtr do begin
            writeln(statOutFile,t:4,
                         qtrSSSMAArry[t].Mean:8:2,
                         qtrSSSMAArry[t].CILow:8:2,
                         qtrSSSMAArry[t].CIHigh:8:2);
          end;
        end;
        close(statOutFile);
    end;
end;

procedure DisplaySimStats (var simACWTBO, simACWT, simSMA, simInvest,
                           simOrderCount, simLastOH, simlastOS,simTotalCost,
                           simACWTBOVar, simACWTVar, simSMAVar,
                           simInvestVar,simOrderCountVar,
                           simLastOHVar,simlastOSVar,simTotalCostVar,
                           simInapp,simInappVar,simInappVal,simInappValVar,
                           simMeanRisk,simMeanRiskVar,simInitSSOH,simInitSSOS,
                           simInitSSOrders,simInitSSOHVar,simInitSSOSVar,
                           simInitSSOrdersVar:real;
                         var n:integer;
                         initInv,initOS,initOrders,
                         initSSOH,initSSOS,initSSOrders:integer;
                         outputType:char; hour1,minute1,second1,hdSec1,
                         hour2,minute2,second2,hdSec2:word);
```

```
var simACWTBOHi,simACWTHi,simSMAHi,simInvestHi,simOrderCountHi,
    simLastOHHi,simLastOSHi,simACWTBOLo,simACWTLo,simSMALo,
    simInvestLo,simOrderCountLo,simLastOHLo,simLastOSLo,
    simTotalCostHi,simTotalCostLo,simInappLo,simInappHi,
    simInappValLo,simInappValHi,simMeanRiskHi,simMeanRiskLo,
    simInitSSOHHi,simInitSSOHLo,simInitSSOSHi,simInitSSOSLo,
    simInitSSOrdersHi,simInitSSOrdersLo:real;


begin

   ConfInv(simACWTBOVar, simACWTBO, simACWTBOHi,simACWTBOLo,n);
   ConfInv(simACWTVar, simACWT, simACWTHi, simACWTLo,n);
   ConfInv(simSMAVar, simSMA, simSMAHi, simSMALo,n);
   ConfInv(simInvestVar, simInvest, simInvestHi, simInvestLo,n);
   ConfInv(simOrderCountVar, simOrderCount, simOrderCountHi,
         simOrderCountLo,n);
   ConfInv(simLastOHVar, simLastOH, simLastOHHi, simLastOHLo,n);
   ConfInv(simLastOSVar, simLastOS, simLastOSHi, simLastOSLo,n);
   ConfInv(simTotalCostVar, simTotalCost, simTotalCostHi, simTotalCostLo,n);
   ConfInv(simInappVar,simInapp, simInappHi, simInappLo,n);
   ConfInv(simInappValVar,simInappVal, simInappValHi, simInappValLo,n);
   ConfInv(simMeanRiskVar,simMeanRisk, simMeanRiskHi, simMeanRiskLo,n);
   ConfInv(simInitSSOHVar, simInitSSOH, simInitSSOHHi, simInitSSOHLo,n);
   ConfInv(simInitSSOSVar, simInitSSOS, simInitSSOSHi, simInitSSOSLo,n);
   ConfInv(simInitSSOrdersVar, simInitSSOrders, simInitSSOrdersHi,
         simInitSSOrdersLo,n);


   writeln
(outputfile,'**************************************************************
************');
   writeln (outputfile,'Init OH Qty: ',initInv:8,
         '  Init SS OH Qty: ',simInitSSOH:8:2,'  (',simInitSSOHLo:0:2,
                                    ',',simInitSSOHHi:0:2,')');
   writeln (outputfile,'Init OS Qty: ',initOS:8,
         '  Init SS OS Qty: ',simInitSSOS:8:2,'  (',simInitSSOSLo:0:2,
                                   ',',simInitSSOSHi:0:2,')');
   writeln (outputfile,'Init Orders: ',initOrders:8,
         '  Init SS Orders: ',simInitSSOrders:8:2,'  (',simInitSSOrdersLo:0:2,
                                     ',',simInitSSOrdersHi:0:2,')');


   writeln(outputfile);
   writeln(outputfile,'Simulation Final Statistics');
   writeln(outputfile);
   writeln(outputfile,'      ACWTBO  ACWT  SMA  Orders INVEST  End OH  End OS');
   writeln(outputfile,'     ',simACWTBO:7:2,simACWT:7:2,simSMA:7:2,simOrderCount:8:2,
         simInvest:8:2,simLastOH:8:2,simLastOS:8:2);
   writeln(outputfile,'Low  ',simACWTBOLo:7:2,simACWTLo:7:2,simSMALo:7:2,
        simOrderCountLo:8:2,simInvestLo:8:2,simLastOHLo:8:2,
```

```
        simLastOSLo:8:2);
  writeln(outputfile,'High  ',simACWTBOHi:7:2,simACWTHi:7:2,simSMAHi:7:2,
        simOrderCountHi:8:2,simInvestHi:8:2,simLastOHHi:8:2,
        simLastOSHi:8:2);
  writeln(outputfile);
  writeln(outputfile,'     Total Cost   INAPP INAPP Value   Mean Risk');
  writeln(outputfile,'
',simTotalCost:10:2,simInapp:8:2,simInappVal:10:2,simMeanRisk:10:2);
  writeln(outputfile,'Low
',simTotalCostLo:10:2,simInappLo:8:2,simInappValLo:10:2,simMeanRiskLo:10:2);
  writeln(outputfile,'High
',simTotalCostHi:10:2,simInappHi:8:2,simInappValHi:10:2,simMeanRiskHi:10:2);

  if n<30 then begin
    writeln(outputfile);
    writeln(outputfile,'Caution! The confidence level is based on a normality assumption.');
    writeln(outputfile,'Your sample has only ',n:3,' values');
  end;
  writeln
(outputfile,'**********************************************************************
************');
  writeln(outputfile,'Sim Start Time   ',hour1,':',minute1,':',second1,':',hdSec1,
              '  Sim End Time     ',hour2,':',minute2,':',second2,':',hdSec2);
  if outputType ='1' then HitToCont;
end;



begin    {main}
  textcolor(14);
  stop:=FALSE;
  simCount:=0;
  Frontscreen;
  Runtype (distrType,outputType,wkDataType,qtrDataType,PDDataType,RunPD86Type,
        repStatType,numberOfQtrs,numberOfWks,numberOfReps,negBinS,

seedIndex,startSSQtr,endSSQtr,meanDemand,varDemand,negBinP,inputfile,outputfile,
        frcst,mad,seeds,outFileName,runDescript);
  repeat
    rewrite (outputfile);
    simCount:=simCount+1;
    currSeed:=0;
    n:=0;
    GetTime( hour1,minute1,second1,hdSec1);
    InitializeStatArrays(qtrACWTBOArry,qtrACWTArry,qtrSMAArry,
                qtrInvestArry,qtrInappArry,cumACWTBOArry,
                cumACWTArry,cumSMAArry);
    for numberRep := 1 to numberOfReps do begin
      if seedType = '1' then begin
        if numberRep = 1 then begin
```

```
    for s:= 1 to seedIndex do begin
      currSeed:= GetNextSeed(currSeed);
    end;
    SetSeed(currSeed);
  end else begin
    currSeed:= GetNextSeed (currSeed);
    SetSeed(currSeed);
  end;
end else begin
  SetSeed(seeds[numberRep]);
end;
InitializeArrays (observ,meanDmdArry,varDmdArry,EOQArry,
            ROLevelArry,APSRArry,attainRisk,
            SSADDBO,SSADD,SSSMA,
            stepIndArry, trndIndArry,mkCodeArry,
            numberOfQtrs,numberOfWks,meanDemand,
            wklyObserv);
LoadObserv (observ,frcst,mad,meanDmdArry,varDmdArry,wklyObserv,
        observType,distrType,numberOfQtrs,numberOfWks,numberRep,
        simCount,trendOn,stepOn,nmbrSteps, nmbrTrends,
        meanDemand,varDemand,inputfile,startstep,
        startrnd, endtrnd,stepmult, trendcoeff, trendpower);
if numberRep = 1 then begin
  if simCount= 1 then InitPD82File (prbBrkPt,PLTSigMuRatio,
                        obsol,timePref,storage,
                        shortCost,adminCost);
  PD82Edit(prbBrkPt,unitPrice,PLT,PLTSigMuRatio,
        obsol,timePref,storage,shortCost,adminCost);
end;
if numberRep= 1 then PrintHeader(prbBrkPt,negBins,seedIndex,
                    meanDemand,varDemand,
                    PLTSigMuRatio,negBinP,
                    outputfile,outputType,distrType,
                    outFileName,runDescript,nmbrSteps,
                    nmbrTrends,startSSQtr,endSSQtr,
                    stepMult,trendCoeff,
                    trendPower,startStep,starTrnd,
                    endTrnd);
Forecast (observ,frcst,mad, stepIndArry, trndIndArry,
      mkCodeArry,numberOfQtrs,numberRep,unitPrice);
LoadLevels (frcst, mad, observ, EOQArry, ROLevelArry,
        APSRArry, attainRisk,
        SSADDBO, SSADD, SSSMA,
        qtrSSADDBOArry,qtrSSADDArry,qtrSSSMAArry,
        mkCodeArry,numberOfQtrs, prbBrkPt, numberRep,
        meanDemand,PLTSigMuRatio,meanRisk,PDDataType,RunPD86Type);
if PDDataType='1' then DisplayPDOutput (observ, frcst, mad, EOQArry,
                        ROLevelArry, APSRArry,
                        attainRisk, SSADDBO, SSADD,
                        SSSMA, stepIndArry, trndIndArry,
```

```
                                mkCodeArry,numberOfQtrs,initInv,
                                numberRep,outputType);

      SDR(OSHeap,BOHeap,wklyObserv,EOQArry,ROLevelArry,observ,numberOfQtrs,
         initInv,initOS,initOrders,startSSQtr,endSSQtr, initSSOH,
         initSSOS,initSSOrders,PLT,meanDemand,
         PLTSigMuratio,obsol,timePref,
         storage,shortCost,adminCost,TWUS,ACWTBO,
         ACWT,SMA,Invest,orderCount,lastOH,lastOS,totalCost,inappAsset,
         inappVal,wkDataType,qtrDataType,outputType,qtrACWTBOArry,qtrACWTArry,
         qtrSMAArry,qtrInvestArry,qtrInappArry,cumACWTBOArry,cumACWTArry,
         cumSMAArry,numberRep);
      if repStatType = '1' then DisplayRepStats (ACWTBO, ACWT, SMA, Invest,
orderCount,
                                lastOH,lastOS,totalCost,
                                inappAsset,inappVal,outputType );
      CalcSimStats(ACWTBO, ACWT, SMA, Invest,orderCount,lastOH,lastOS,
            totalCost,inappAsset,inappVal,meanRisk,
            initSSOH, initSSOS,initSSOrders,n,
            simACWTBO,simACWT,simSMA,simInvest,
            simOrderCount,simLastOH,simLastOS,
            simTotalCost,simACWTBOVar,simACWTVar,
            simSMAVar,simInvestVar,simOrderCountVar,
            simLastOHVar,simLastOSVar,
            simTotalCostVar,simInapp,
            simInappVar,simInappVal,simInappValVar,simMeanRisk,
            simMeanRiskVar,simInitSSOH,simInitSSOS,simInitSSOrders,
            simInitSSOHVar,simInitSSOSVar,simInitSSOrdersVar);
   end; {for}
   GetTime(hour2,minute2,second2,hdSec2);
   DisplaySimStats( simACWTBO,simACWT,simSMA,simInvest,simOrderCount,
            simLastOH,simLastOS,simTotalCost,
            simACWTBOVar, simACWTVar,
            simSMAVar, simInvestVar,simOrderCountVar,
            simLastOHVar,simlastOSVar,simTotalCostVar,
            simInapp,simInappVar,simInappVal,simInappValVar,
            simMeanRisk,simMeanRiskVar,
            simInitSSOH,simInitSSOS,simInitSSOrders,
            simInitSSOHVar,simInitSSOSVar,simInitSSOrdersVar,
            n,initInv,initOS,initOrders,
            initSSOH,initSSOS,initSSOrders,
            outputType,hour1,minute1,
            second1,hdSec1,hour2,minute2,second2,hdSec2);
   close (outputfile);
   DisplayQtrSimStats (qtrACWTBOArry,qtrACWTArry,qtrSMAArry,
            qtrInvestArry,qtrInappArry, cumACWTBOArry,cumACWTArry,
            cumSMAArry,qtrSSADDBOArry,qtrSSADDArry,
            qtrSSSMAArry,numberOfReps,numberOfQtrs,
            runDescript,RunPD86Type,startSSQtr,endSSQtr);
   RunAgain (outputfile,runDescript,outputType,
```

```
            frcst,mad,stop,outFileName);
  until stop;
  textcolor(15);
  clrscr;
end. {main}
```

```
(*************************************************************************
***
*This Unit provides a toolbox of useful functions functions and        *
*procedures for data input.                                            *
*************************************************************************
***)

unit toolbox;

Interface

Uses CRT;

type pd82field=string[15];

var  strTemp:pd82field;

function Get_Answer:boolean;
procedure HitToCont;
function Get_Integer (low,high:integer):integer;
function Get_Real(low,high:real):real;
function NumToString (var value:real):pd82field;
function StringToReal (var S:pd82field):real;
function Get_LongInt (low,high:longint):longint;
function NewMean(var mean, sample:real; n:integer):real;
function NewVar(var mean, oldMean, oldVar, sample:real; n:integer):real;
procedure ConfInv(currVar, currMean:real;var upper, lower:real; n:integer);

Implementation

const ERROR=1.00000000000000E-0010;

function Get_Answer;          {Returns a Boolean result for a yes/no query}

var Char_In:Char;
   Correct:Boolean;

begin
  Correct:=False;
  repeat
    Char_In:=ReadKey;
    write (Char_In);
    case Char_In of
      'Y','y':begin
            writeln ('es');
            Get_Answer:=True;
            Correct:=True
      end;
      'N','n', chr(13):begin
            if (Char_In = 'N') or (Char_In = 'n') then begin
```

```
            writeln ('o');
          end else begin
            writeln;
            write ('No');
          end;
          Get_Answer:=False;
          Correct:=True
      end;
      else  begin
          writeln;
          Sound(220);
          delay (300);
          NoSound;
          writeln ('** Un-recognizable answer **');
          writeln ('Enter Y or N,');
          writeln ('Re-enter your answer: ')
      end
    end; {case}
  until Correct;
  end; {Get_Answer}

procedure HitToCont;

var dummy:char;

begin
writeln;
write ('                Hit any key to continue ....');
dummy:=readkey;
end;


{Gets an integer input between low and high, prompts until one is received}
function Get_Integer (low,high:integer):integer;

var numberString: string[10];
   error, numberValue: integer;


begin
  repeat
    readln (numberString);
    val (numberString, numberValue, error);
    if error <> 0 then begin
      writeln;
      Sound(220);
      delay (300);
      NoSound;
      write ('*** Invalid number, enter an integer: ')
    end else if (numberValue<low) or (numberValue>high) then begin
```

```
      writeln;
      Sound(220);
      delay (300);
      NoSound;
      writeln ('*** Invalid Range - value must be a positive integer');
      write ('between ',low,' and ',high,' Enter number: ');
      error:=1;
      end;
  until error=0;
  Get_Integer:=numberValue;
end; {function}

{Gets an longint input between low and high, prompts until one is received}
function Get_LongInt (low,high:longint):longint;

var numberString: string[10];
    error: integer;
    numberValue: longint;

begin
  repeat
     readln (numberString);
     val (numberString, numberValue, error);
     if error <> 0 then begin
        writeln;
        Sound(220);
        delay (300);
        NoSound;
        write ('*** Invalid number, enter an integer: ')
     end else if (numberValue<low) or (numberValue>high) then begin
        writeln;
        Sound(220);
        delay (300);
        NoSound;
        writeln ('*** Invalid Range - value must be a positive integer');
        write ('between ',low,' and ',high,' Enter number: ');
        error:=1;
        end;
  until error=0;
  Get_LongInt:=numberValue;
end; {function}


{Gets a real value between low and high, prompts until one is received}
function Get_Real(low,high:real):real;

var Number_String:string;
    Error:integer;
    Number_Value:real;
```

```
begin
 repeat
   readln (Number_String);
   val (Number_String, Number_Value, Error);
   if Error <> 0 then begin
        Sound(220);
        delay (300);
        NoSound;
        writeln ('**You must enter a valid real number** ');
     end else if (Number_Value<low) or (Number_Value>high) then begin
        writeln;
        Sound(220);
        delay (300);
        NoSound;
        writeln ('*** Invalid Range - value must be a real value');
        write ('between ',low:0:2,' and ',high:0:2,' Enter number: ');
        error:=1;
      end;
 until Error=0;
 Get_Real:=Number_Value;
end; {Get_Real}

function NumToString (var value:real):pd82field;

const digits = 16;
    decimals = 8;

var i:integer;
   S: string[16];
begin
  str (value:digits:decimals,S);
  for i:=1 to 16 do
   if S[i] = ' ' then S[i]:='0'
     else if S[i] = '.' then delete (S,i,1);
NumToString:= S
end;

function StringToReal (var S:pd82field):real;

var  R1, R2: real;
    S1:string[7];
    S2:string[8];
    error1, error2:integer;

begin
S1:=copy(S,1,7);
S2:=copy(S,8,8);
val(S1,R1,error1);
val(S2,R2,error2);
StringToReal:=R1+(R2/100000000);
```

```
end;

function NewMean(var mean, sample:real; n:integer):real;

var calcMean:real;

begin
  if n<1 then begin
    NewMean:=sample;
  end else begin
    calcMean:=(((n-1)*mean)+sample)/n;
    if calcMean < ERROR then begin
      NewMean:=0.0;
    end else begin
      NewMean:=calcMean;
    end;
  end;

end;

function NewVar(var mean, oldMean, oldVar, sample:real; n:integer):real;

var calcVar:real;

begin
  if n < 2.0 then begin
    NewVar:=0.0;
  end else begin
    calcVar:=(((n-2)*oldVar)+((n-1)*sqr(oldMean))-
          (n*sqr(mean))+(sqr(sample)))/(n-1);
    if calcVar < ERROR then begin
      NewVar:=0.0;
    end else begin
      NewVar:=calcVar;
    end;
  end;
end;

procedure ConfInv(currVar, currMean:real;var upper, lower:real; n:integer);

begin
  if (n>0) and (currVar > ERROR) then begin
    lower:=currMean-(1.96*sqrt(currVar/n));
    upper:=currMean+(1.96*sqrt(currVar/n));
  end else begin
    lower:=0.0;
    upper:=0.0;
  end;
  if lower < 0.0 then lower:=0.0;
  if upper < 0.0 then upper:=0.0;
```

```
end;

end. {Unit Toolbox}
```

```
unit unirand;

interface

type seedArryType = array  [1..100] of longint;

var seeds:seedArryType;

procedure SetSeed (seed:longint);

function GetSeed:longint;

function GetNextSeed (lastSeed:longint):longint;

function RandomUniform:real;

function GetPoisson(var meanDemand:real):integer;

function GetNormal:real;

function GetGeometric(p:real):integer;

function GetNegBin(p:real;s:integer):integer;

function GetUniformInt(high:integer):integer;

function ZInv (p:real):real;

implementation

var a:longint;

procedure SetSeed (seed:longint);

begin
  a:=seed
end; {procedure}


function GetSeed:longint;

begin
  GetSeed:=a
end; {procedure}


function RandomUniform:real;

const B2E15:longint=32768;
    B2E16:longint=65536;
```

```
    Modlus:longint=2147483647;
    Mult1:longint=24112;
    Mult2:longint=26143;

var Hi15,Hi31,Low15,Lowprd,Ovflow,Zi:longint;

begin
  Zi:=a;
  Hi15:=Zi div B2E16;
  Lowprd:=(Zi - Hi15 * B2E16) * Mult1;
  Low15:=Lowprd div B2E16;
  Hi31:=Hi15 * Mult1 + Low15;
  Ovflow:=Hi31 div B2E15;
  Zi:=(((Lowprd - Low15 * B2E16) - Modlus) +
      (Hi31 - Ovflow * B2E15) * B2E16) + Ovflow;
  if Zi < 0 then Zi:= Zi + Modlus;
  Hi15:= Zi div B2E16;
  Lowprd:= (Zi - Hi15 * B2E16) * Mult2;
  Low15:=Lowprd div B2E16;
  Hi31:= Hi15 * Mult2 + Low15;
  Ovflow:= Hi31 div B2E15;
  Zi:=(((Lowprd - Low15 * B2E16) - Modlus) +
      (Hi31 - Ovflow * B2E15) * B2E16) + Ovflow;
  if Zi < 0 then Zi:= Zi + Modlus;
  a:=Zi;
  RandomUniform:= (2 * (Zi div 256) + 1) / 16777216.0;
end;

function GetNextSeed (lastSeed:longint):longint;

const M:extended=2147483647.0;
    a:extended=715.0;
    b:extended=1058.0;
    c:extended=1385.0;

var  Z:extended;

begin
  Z:=lastSeed;
  if lastSeed=0 then begin
    Z:=1973272912.0;
    GetNextSeed:=round(Z);
  end else begin
    Z:=(A*Z) / M;
    Z:=(Z-round(Z-0.5))*M;
    Z:=(B*Z) / M;
    Z:=(Z-round(Z-0.5))*M;
    Z:=(C*Z) / M;
    Z:=(Z-round(Z-0.5))*M;
    GetNextSeed:=round(Z);
```

```
  end;
end;

function GetPoisson(var meanDemand:real):integer;

var alpha,beta, U1:real;
   i:integer;

begin
  beta:=1.0;
  i:=-1;
  repeat
    i:=i+1;
    alpha:= exp(-meanDemand);
    U1:=RandomUniform;
    beta:=beta*U1;
  until beta<alpha;
  GetPoisson:=i
end;

function GetNormal:real;

var U1,U2,V1,V2,W,Y:real;

begin
  repeat
    U1:=RandomUniform;
    U2:=RandomUniform;
    V1:=2*U1-1; V2:=2*U2-1;
    W:=sqr(V1)+sqr(V2);
  until W <= 1.0;
  Y:=sqrt((-2*ln(W))/W);
  GetNormal:=V1*Y;
end;

function GetGeometric(p:real):integer;

var U:real;
   i:integer;

begin
  i:=0;
  U:=RandomUniform;
  while not(U <= p) do begin
    i:=i+1;
    U:=RandomUniform;
  end;
  GetGeometric:=i;
end;
```

```
function GetNegBin(p:real;s:integer):integer;

var X,i:integer;

begin
  X:=0;
  for i:=1 to s do begin
     X:=X+GetGeometric(p);
  end;
  GetNegBin:=X;
end;

function GetUniformInt(high:integer):integer;

begin
  GetUniformInt:=round((high-1)*RandomUniform)+1;
end;

function ZInv (p:real):real;

var t:real;

begin
  t:=sqrt(-2*ln(p));
  ZInv:=t-((2.515517+0.802853*t+0.010328*sqr(t))/
        (1+1.432788*t+0.189269*sqr(t)+0.001308*exp(3*ln(t))));
end;

end. {unit unirand}
```

```
unit pQueue;

interface

const MAXPQUEUESIZE=300;

type dataRecord = record
                  Qty:integer;
                  Week:integer;
                  end;
    HeapArrayType=array [1..MAXPQUEUESIZE] of datarecord;
    PriorityQueueType = record
      heapSize:integer;
      heapArray:HeapArrayType
    end;

{must be called before the priority queue is first used}
{also resets the priority queue so it is empty}
procedure InitializePriorityQueue (var pQueue:PriorityQueueType);

{error if called when it already has MAXPQUEUESIZE elements}
procedure InsertPriorityQueue (var pQueue:PriorityQueueType; data:datarecord);

{returns the element with the smallest (next time) value}
{error if no elements in the priority queue}
function CurrWeek (pQueue:PriorityQueueType):integer;
function CurrQty (pQueue:PriorityQueueType):integer;

{removes and returns the element with the smallest (next time) value}
{error if no elements in the priority queue}
function ExtractQty (var pQueue:PriorityQueueType):integer;
function ExtractWeek (var pQueue:PriorityQueueType):integer;

function EmptyPriorityQueue (pQueue:PriorityQueueType):boolean;

function SizePriorityQueue (pQueue:PriorityQueueType):integer;

implementation

{error if the binary trees that are children of the index do not satisfy the
heap property}
procedure Heapify (var pQueue:PriorityQueueType; i:integer);

var left,right,smallest:integer;
   tempVar:dataRecord;

begin
  with pQueue do begin
    left:=2*i;
    right:=(2*i)+1;
```

```
    smallest:=i;
    if (left <= heapSize) then begin
      if (heapArray [left].Week < heapArray[i].Week) then begin
        smallest:=left
      end
    end;
    if (right<=heapSize) then begin
      if (heapArray[right].Week < heapArray[smallest].Week) then begin
        smallest:=right
      end
    end;
    if smallest <>i then begin
      tempVar:=heapArray[i];
      heapArray[i]:=heapArray[smallest];
      heapArray[smallest]:=tempVar;
      Heapify (pQueue,smallest)
    end
  end {with}
end; {procedure}


{removes and returns the element with the smallest (next time) value}
{error if no elements in the priority queue}
function HeapExtractWeek (var pQueue:PriorityQueueType):integer;

begin
  with pQueue do begin
    HeapExtractWeek:=heapArray[1].Week;
    heapArray[1]:=heapArray[heapSize];
    heapSize:=heapSize-1;
    Heapify (pQueue,1)
  end {with}
end; {procedure}

{removes and returns the element with the smallest (next time) value}
{error if no elements in the priority queue}
function HeapExtractQty (var pQueue:PriorityQueueType):integer;

begin
  with pQueue do begin
    HeapExtractQty:=heapArray[1].Qty;
    heapArray[1]:=heapArray[heapSize];
    heapSize:=heapSize-1;
    Heapify (pQueue,1)
  end {with}
end; {procedure}


{error if called when it already has MAXPQUEUESIZE elements}
procedure HeapInsert (var pQueue:PriorityQueueType; data:datarecord);
```

```
var index, parent:integer;
   done:boolean;

begin
  with pQueue do begin
    done:=false;
    heapSize:=heapSize+1;
    index:=heapSize;
    parent:=index div 2;
    if parent=0 then begin
        done:=TRUE
    end else if (heapArray[parent].Week <= data.Week) then begin
        done:=TRUE
    end;
    while (index > 1) and (not done) do begin
      heapArray[index]:=heapArray[parent];
      index:=parent;
      parent:=index div 2;
      if parent=0 then begin
        done:=TRUE
      end else if (heapArray[parent].Week <= data.Week) then begin
        done:=TRUE
      end
    end; {while}
  heapArray[index]:=data
  end {with}
end; {procedure}


procedure InitializePriorityQueue (var pQueue:PriorityQueueType);

var index:integer;

begin
  pQueue.heapSize:=0
end;  {procedure}


procedure InsertPriorityQueue (var pQueue:PriorityQueueType; data:dataRecord);

begin
  HeapInsert (pQueue, data)
end; {procedure}


function CurrWeek (pQueue:PriorityQueueType):integer;

begin
  CurrWeek:=pQueue.heapArray[1].Week;
end; {function}
```

```
function CurrQty (pQueue:PriorityQueueType):integer;

begin
  CurrQty:=pQueue.heapArray[1].Qty;
end;  {function}


function ExtractQty (var pQueue:PriorityQueueType):integer;

begin
  ExtractQty:=HeapExtractQty (pQueue)
end;  {function}

function ExtractWeek (var pQueue:PriorityQueueType):integer;

begin
  ExtractWeek:=HeapExtractWeek (pQueue)
end;  {function}


function EmptyPriorityQueue (pQueue:PriorityQueueType):boolean;

begin
  EmptyPriorityQueue:=pQueue.heapSize=0
end;   {function}


function SizePriorityQueue (pQueue:PriorityQueueType):integer;

begin
  SizePriorityQueue:=pQueue.heapSize
end;  {function}

end. {unit}
```

```
unit PDUnit;

Interface

uses dos, crt, toolbox;

var prbBrkPt :integer;
   unitPrice, PLT, obsol, timePref, storage, shortCost,adminCost:real;

procedure InitPD82File (var prbBrkPt:integer;var PLTSigMuRatio,
                  obsol,timePref,storage,shortCost,adminCost:real);

procedure PD82Edit(var prbBrkPt:integer; var unitPrice,PLT,PLTSigMuRatio,
             obsol,timePref,storage,shortCost,adminCost:real);

procedure InitPD86File;

Implementation

procedure InitPD82File (var prbBrkPt:integer;var PLTSigMuRatio,
                  obsol,timePref,storage,shortCost,adminCost:real);

var AAC,AL,B067A,B067G,C028,DRLI,DO31C,D125N,ERRI,F024,HQDI,MARLI,PVPI,RII,RO,
   YR7POC,Y006A,Y006B,EOQIND,PVUI : char;

   D120, FILLER : string [2];

   A023B,BRLDC,B010,B011A,B012F,B019A,B020,B023C,B023D,B023F,B023H,BG,B055,
   B055A,B057,B058,B058A,B061,B070,B073,B093,B280,C008C,DOPTC,DTC,D025E,
   F009,HQD,H0141,H0142,H0143,H0144,H0145,H0146,H0147,H0148,H0149,H01410,
   H01411,H01412,H01413,H01414,H01415,H01416,H01417,H01418,H01419,H01420,

ILR,IMECY,M,MOQQAD,MSLQAD,MSLQD,NRFIDRT,OSQ,PDQ,PPV,QDH,RFIDRT,RIYAYABY
,
   RSV,RT,SCR,SSOH,TD,TSDRS,V015R,V016,V022,V039,V041R,V042R,V043R,V044,

V101A,V102,V1034,V108,V295,LILT,LILY,PCR3,Q1B,Q2B,RMNAST,SER,YDR,MNQQAD,
   APSR,ARCI,BOQ,BRLCI,BRLDCU,BRLQ,BRPLQ,BRQ,B014A,B019,B019B,B021,B021A,
   ERR,MONDO,OQCI,POC,PPVBNDO,PZO,RCI,RLCI,RPLCI,RQCI,VPSR : real;

   PD82str1: string[24];
   PD82str2, PD82str3, PD82str4, PD82str5, PD82str6, PD82str7,
   PD82str8: string[255];

   outfile:text;

begin

{initialization values}
AAC:='N'; AL:='N'; B067A:='N'; B067G:='N'; C028:='0'; DRLI:='N'; DO31C:=' ';
```

```
D120:='06'; D125N:=' '; ERRI:='N'; F024:=' '; HQDI:=' '; MARLI:='Y';
PVPI:='Y'; RII:='N'; RO:='N'; YR7POC:= ' '; Y006A:='N'; Y006B:='N';
EOQIND:='N'; PVUI:=' '; FILLER:='  ';
A023B:= 1.0;          {system requisition average}
BRLDC:=5.0;           {basic reorder level distribution code}
B010:=0.0;            {contract prod lead time}
B011A:=8.0;           {contract proc lead time}
B012F:=0.0;
B019A:=20.0;          {non cred group proc variance}
B020:=1.0;            {system reorder level low limit qty}
B023D:=1.0;           {gross sys demand end of lead time}
B023C:=B011A*B023D;    {gross sys demand during lead time}
B023F:=0.0; B023H:=0.0; BG:=0.0;
B055:=100.00;         {unit price}
B055A:=0.0;
B057:=0.12;           {obsolescence rate}
B058:=600.0;          {manufac set-up costs}
B058A:=0.0;
B061:=1.0;            {discount rate}
B070:=0.0;
B073:=1.0;            {expected units per requisition}
B093:=0.0; B280:=0.0;
C008C:=0.5;           {average item essentiality}
DOPTC:=0.0;DTC:=0.0;
D025E:=0.0;           {procurement method}
F009:=0.0; HQD:=0.0; H0141:=0.0; H0142:=0.0; H0143:=0.0; H0144:=0.0;
H0145:=0.0; H0146:=0.0; H0147:=0.0; H0148:=0.0; H0149:=0.0; H01410:=0.0;
H01411:=0.0; H01412:=0.0; H01413:=0.0; H01414:=0.0; H01415:=0.0;
H01416:=0.0;
H01417:=0.0; H01418:=0.0; H01419:=0.0; H01420:=0.0; ILR:=0.0; IMECY:=0.0;
M:=1.0;               {mark code}
MOQQAD:=6.0;          {max order qty attrition qtrs demand}
MSLQAD:=99.0;         {max number safety level qtrs attrition}
MSLQD:=20.0;          {max number of safety level qtrs demand}
NRFIDRT:=0.0;
OSQ:=0.0;             {non-parametric order stat qtrs}
PDQ:=8.0;             {past qtrs demand}
PPV:=B023D*B011A;      {proc problem var (mean)}
QDH:=0.0;             {quarters demand history}
RFIDRT:=0.0; RIYAYABY:=0.0;
RSV:=0.0;             {requisition size variance}
RT:=0.0;
SCR:=0.01;            {storage cost rate}
SSOH:=0.0;
TD:=93001.0;          {today's date}
TSDRS:=0.08;          {time between SDR's in qtrs}
V015R:=850.00;        {mark code 1 and 2 order costs}
V016:=850.00;
V022:=0.1;            {min risk}
V039:=0.0;
```

```
V041R:=850.00;          {low value annual demand order cost}
V042R:=1920.00;         {negotiated procurement order cost}
V043R:=1790.00;         {advertised procurement order costs}
V044:=8000.00;          {max unpriced order cost}
V101A:=0.1;             {procurement interest rate}
V102:=0.35;             {max risk}
V1034:=1000.00;         {shortage cost}
V108:=0.1;              {repair time preference rate}
V295:=1.0;              {reorder level constraint}
LILT:=0.0; LILY:=0.0; PCR3:=0.0; Q1B:=0.0; Q2B:=0.0; RMNAST:=0.0; SER:=0.0;
YDR:=0.0;
MNQQAD:=1.0;            {min order qty attrition qtrs demand}
APSR:=0.0; ARCI:=0.0; BOQ:=0.0; BRLCI:=0.0; BRLDCU:=0.0; BRLQ:=0.0;
BRPLQ:=0.0; BRQ:=0.0; B014A:=0.0; B019:=0.0; B019B:=0.0; B021:=0.0;
B021A:=0.0; ERR:=0.0; MONDO:=0.0; OQCI:=0.0; POC:=0.0; PPVBNDO:=0.0;
PZO:=0.0; RCI:=0.0; RLCI:=0.0; RPLCI:=0.0; RQCI:=0.0; VPSR:=0.0;

prbBrkPt:=0;
PLTSigMuRatio:=(sqrt(1.57*B011A))/B011A;
obsol:=B057;
timePref:=V101A;
storage:=SCR;
shortCost:=V1034;
adminCost:=V015R;

pd82str1:= AAC+ AL+ B067A+ B067G+ C028+ DRLI+ DO31C+ D120+ D125N+
ERRI+ F024+
        HQDI+ MARLI+ PVPI+ RII+ RO+ YR7POC+ Y006A+ Y006B+ EOQIND+
PVUI+
        FILLER;

PD82str2:= NumToString(A023B)+ NumToString(BRLDC)+ NumToString(B010)+
        NumToString(B011A)+ NumToString(B012F)+ NumToString(B019A)+
        NumToString(B020)+ NumToString(B023C)+ NumToString(B023D)+
        NumToString(B023F)+ NumToString(B023H)+ NumToString(BG)+
        NumToString(B055)+ NumToString(B055A)+ NumToString(B057)+
        NumToString(B058)+ NumToString(B058A);
PD82str3:= NumToString(B061)+ NumToString(B070)+ NumToString(B073)+
        NumToString(B093)+ NumToString(B280)+ NumToString(C008C)+
        NumToString(DOPTC)+ NumToString(DTC)+ NumToString(D025E)+
        NumToString(F009)+ NumToString(HQD)+ NumToString(H0141)+
        NumToString(H0142)+ NumToString(H0143)+ NumToString(H0144)+
        NumToString(H0145)+ NumToString(H0146);
PD82str4:= NumToString(H0147)+ NumToString(H0148)+ NumToString(H0149)+
        NumToString(H01410)+ NumToString(H01411)+ NumToString(H01412)+
        NumToString(H01413)+ NumToString(H01414)+ NumToString(H01415)+
        NumToString(H01416)+ NumToString(H01417)+ NumToString(H01418)+
        NumToString(H01419)+ NumToString(H01420)+ NumToString(ILR)+
        NumToString(IMECY)+ NumToString(M);
PD82str5:= NumToString(MOQQAD)+ NumToString(MSLQAD)+ NumToString(MSLQD)+
```

```
        NumToString(NRFIDRT) + NumToString(OSQ) + NumToString(PDQ) +
        NumToString(PPV) + NumToString(QDH) + NumToString(RFIDRT) +
        NumToString(RIYAYABY) + NumToString(RSV) + NumToString(RT) +
        NumToString(SCR) + NumToString(SSOH) + NumToString(TD) +
        NumToString(TSDRS) + NumToString(V015R);
PD82str6:= NumToString(V016) + NumToString(V022) + NumToString(V039) +
        NumToString(V041R) + NumToString(V042R) + NumToString(V043R) +
        NumToString(V044) + NumToString(V101A) + NumToString(V102) +
        NumToString(V1034) + NumToString(V108) + NumToString(V295) +
        NumToString(LILT) + NumToString(LILY) + NumToString(PCR3) +
        NumToString(Q1B) + NumToString(Q2B);
PD82str7:= NumToString(RMNAST) + NumToString(SER) + NumToString(YDR) +
        NumToString(MNQQAD) + NumToString(APSR) + NumToString(ARCI) +
        NumToString(BOQ) + NumToString(BRLCI) + NumToString(BRLDCU) +
        NumToString(BRLQ) + NumToString(BRPLQ) + NumToString(BRQ) +
        NumToString(B014A) + NumToString(B019) + NumToString(B019B) +
        NumToString(B021) + NumToString(B021A);
PD82str8:= NumToString(ERR) + NumToString(MONDO) + NumToString(OQCI) +
        NumToString(POC) + NumToString(PPVBNDO) + NumToString(PZO) +
        NumToString(RCI) + NumToString(RLCI) + NumToString(RPLCI) +
        NumToString(RQCI) + NumToString(VPSR);
assign (outfile,'c:\tp\pd82in.fil');
rewrite (outfile);
writeln(outfile,PD82str1, PD82str2, PD82str3, PD82str4, PD82str5, PD82str6,
      PD82str7, PD82str8);
close (outfile);

end;


procedure PD82Edit(var prbBrkPt:integer; var unitPrice,PLT,PLTSigMuRatio,
              obsol,timePref,storage,shortCost,adminCost:real);

var C028 : string[1];

   A023B,B010,B011A,B020,B023C,B023D,B055,B057,B058,B061,B073,C008C,D025E,
   MSLQD,SCR,TD,TSDRS,V015R,V022,V101A,V102,V1034,V295: real;

   PD82str1: string[24];
   PD82str2, PD82str3, PD82str4, PD82str5, PD82str6, PD82str7,
   PD82str8: string[255];
   editChoice:char;
   done:boolean;
   infile,outfile:text;

begin

{retrieve selected default variables from file to edit}
  assign (infile,'c:\tp\pd82in.fil');
  reset (infile);
```

```
  read(infile,PD82str1, PD82str2, PD82str3, PD82str4, PD82str5, PD82str6,
     PD82str7, PD82str8);
  close (infile);
  C028:=copy(PD82str1,5,1);

  strTemp:=copy(PD82str2,31,15);  B010:=StringToReal(StrTemp);
  strTemp:=copy(PD82str2,46,15);  B011A:=StringToReal(StrTemp);
  strTemp:=copy(PD82str2,91,15);  B020:=StringToReal(StrTemp);
  strTemp:=copy(PD82str2,121,15); B023D:=StringToReal(StrTemp);
  strTemp:=copy(PD82str2,181,15); B055:=StringToReal(StrTemp);
  strTemp:=copy(PD82str2,211,15); B057:=StringToReal(StrTemp);
  strTemp:=copy(PD82str2,226,15); B058:=StringToReal(StrTemp);
  strTemp:=copy(PD82str3,1,15);   B061:=StringToReal(StrTemp);
  strTemp:=copy(PD82str3,31,15);  B073:=StringToReal(StrTemp);
  strTemp:=copy(PD82str3,76,15);  C008C:=StringToReal(StrTemp);
  strTemp:=copy(PD82str3,121,15); D025E:=StringToReal(StrTemp);
  strTemp:=copy(PD82str5,31,15);  MSLQD:=StringToReal(StrTemp);
  strTemp:=copy(PD82str5,181,15); SCR:=StringToReal(StrTemp);
  strTemp:=copy(PD82str5,211,15); TD:=StringToReal(StrTemp);
  strTemp:=copy(PD82str5,226,15); TSDRS:=StringToReal(StrTemp);
  strTemp:=copy(PD82str5,241,15); V015R:=StringToReal(StrTemp);
  strTemp:=copy(PD82str6,16,15);  V022:=StringToReal(StrTemp);
  strTemp:=copy(PD82str6,106,15); V101A:=StringToReal(StrTemp);
  strTemp:=copy(PD82str6,121,15); V102:=StringToReal(StrTemp);
  strTemp:=copy(PD82str6,136,15); V1034:=StringToReal(StrTemp);
  strTemp:=copy(PD82str6,166,15); V295:=StringToReal(StrTemp);
  unitPrice:=B055;
  PLT:= B011A;
  done:=FALSE;
  repeat
  clrscr;
  writeln('  **** THIS SCREEN ALLOWS EDITING OF DEFAULT NIIN INPUT PARAMETERS
****');
  writeln;
  writeln;
  writeln (' A.  Prob Break: ',PrbBrkPt:8, '   L.  Min Risk  : ',V022:8:2);
  writeln (' B.  Shelf Life:      ',C028,'   M.  Max Risk  : ',V102:8:2);
  writeln (' C.  Reqn Size : ',B073:8:0, '   N.  Ord Cost  : ',V015R:8:2);
  writeln (' D.  Unit Price: ',B055:8:2,  '   O.  MSLQD     : ',MSLQD:8:2);
  writeln (' E.  Procur LT : ',B011A:8:2,  '   P.  Proc Meth : ',D025E:8:0);
  writeln (' F.  PLT Sig/Mu: ',PLTSigMuRatio:8:2, '   Q.  Shortage  : ',V1034:8:2);
  writeln (' G.  Essential : ',C008C:8:2,  '   R.  R/O Low   : ',B020:8:2);
  writeln (' H.  Mfg Set-Up: ',B058:8:2,  '   S.  R/O Constr: ',V295:8:2);
  writeln (' I.  Obsol Rate: ',B057:8:2,  '   T.  Stor Rate : ',SCR:8:2);
  writeln (' J.  Disc Rate : ',B061:8:2,  '   U.  Time Pref : ',V101A:8:2);
  writeln (' K.  Time SDRS : ',TSDRS:8:2, '   V.  Today DT  : ',TD:8:0);

  writeln;
  writeln ('    Hit ENTER to accept current values ');
  write ('      or letter of field to change: ');
```

```
editChoice:=upcase(readkey);
writeln(editChoice);
case editChoice of
    'A' : begin
            writeln;
            write ('Enter new Probability Break Point: ');
            PrbBrkPt:=Get_Integer(0,20);
          end;
    'B' : begin
            writeln;
            write ('Enter new Shelf Life code: ');
            readln (C028);
            delete (PD82str1,5,1);
            insert (C028,PD82str1,5);
          end;
    'C' : begin
            writeln;
            writeln ('** Information Only - Model assumes requisition size of one. **');
            HitToCont;
          end;
    'D' : begin
            writeln;
            write ('Enter new Unit Price: ');
            B055:=Get_Real(0.0,999999.0);
            delete (PD82str2,181,15);
            insert (NumToString(B055),PD82str2,181);
            unitPrice:=B055;
          end;
    'E' : begin
            writeln;
            write ('Enter new Procurement Leadtime Forecast: ');
            B011A:=Get_Real(0.1,16.0);
            B023C:=B011A*B023D;
            delete (PD82str2,46,15);
            insert (NumToString(B011A),PD82str2,46);
            delete (PD82str2,106,15);
            insert (NumToString(B023C),PD82str2,106);
            PLT:=B011A;
            PLTSigMuRatio:=(sqrt(1.57*B011A))/B011A;
          end;
    'F' : begin
            writeln;
            write ('Enter new PLT Std Deviation to Mean Ratio: ');
            PLTSigMuRatio:=Get_Real(0.0,3.0);
          end;
    'G' : begin
            writeln;
            write ('Enter new Average Item Essentiality: ');
            C008C:=Get_Real(0.0,999999.0);
            delete (PD82str3,76,15);
```

```
        insert (NumToString(C008C),PD82str3,76);
      end;
'H' : begin
        writeln;
        write ('Enter new Manufacturer Set-up Cost: ');
        B058:=Get_Real(0.0,999999.0);
        delete (PD82str2,226,15);
        insert (NumToString(B058),PD82str2,226);
      end;
'I' : begin
        writeln;
        write ('Enter new Obsolescence Rate: ');
        B057:=Get_Real(0.0,999999.0);
        delete (PD82str2,211,15);
        insert (NumToString(B057),PD82str2,211);
        obsol:=B057;
      end;
'J' : begin
        writeln;
        write ('Enter new Discount Rate: ');
        B061:=Get_Real(0.0,999999.0);
        delete (PD82str3,1,15);
        insert (NumToString(B061),PD82str3,1);
      end;
'K' : begin
        writeln;
        writeln ('      ***** Information Only *****');
        delay(1000);
       { write ('Enter new Time Between SDRs: ');
        TSDRS:=Get_Real(0.0,999999.0);
        delete (PD82str5,226,15);
        insert (NumToString(TSDRS),PD82str5,226);}
      end;
'L' : begin
        writeln;
        write ('Enter new Minimum Risk: ');
        V022:=Get_Real(0.0,1.0);
        delete (PD82str6,16,15);
        insert (NumToString(V022),PD82str6,16);
      end;
'M' : begin
        writeln;
        write ('Enter new Maximum Risk: ');
        V102:=Get_Real(0.0,1.0);
        delete (PD82str6,121,15);
        insert (NumToString(V102),PD82str6,121);
      end;
'N' : begin
        writeln;
        write ('Enter new Mark I/II Order Cost: ');
```

```
        V015R:=Get_Real(0.0,999999.0);
        delete (PD82str5,241,15);
        insert (NumToString(V015R),PD82str5,241);
        adminCost:=V015R;
      end;
'O' : begin
        writeln;
        write ('Enter new Max Number of Quarters Safety Level Demand: ');
        MSLQD:=Get_Real(0.0,999999.0);
        delete (PD82str5,31,15);
        insert (NumToString(MSLQD),PD82str5,31);
      end;
'P' : begin
        writeln;
        write ('Enter new Procurement Method: ');
        D025E:=Get_Real(0.0,999999.0);
        delete (PD82str3,121,15);
        insert (NumToString(D025E),PD82str3,121);
      end;
'Q' : begin
        writeln;
        write ('Enter new Procurement Shortage Cost: ');
        V1034:=Get_Real(0.0,999999.0);;
        delete (PD82str6,136,15);
        insert (NumToString(V1034),PD82str6,136);
        shortCost:=V1034;
      end;
'R' : begin
        writeln;
        write ('Enter new System Reorder Level Low Limit Qty: ');
        B020:=Get_Real(0.0,999999.0);
        delete (PD82str2,91,15);
        insert (NumToString(B020),PD82str2,91);
      end;
'S' : begin
        writeln;
        write ('Enter new Reorder Level Constraint Rate: ');
        V295:=Get_Real(0.0,999999.0);
        delete (PD82str6,166,15);
        insert (NumToString(V295),PD82str6,166);
      end;
'T' : begin
        writeln;
        write ('Enter new Storage Cost Rate: ');
        SCR:=Get_Real(0.0,99999.0);
        delete (PD82str5,181,15);
        insert (NumToString(SCR),PD82str5,181);
        storage:=SCR;
      end;
'U' : begin
```

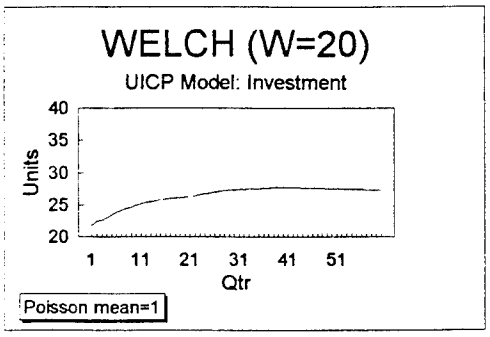
```
            writeln;
            write ('Enter new Time Preference Rate: ');
            V101A:=Get_Real(0.0,99999.0);
            delete (PD82str6,106,15);
            insert (NumToString(V101A),PD82str6,106);
            timePref:=V101A;
          end;
      'V' : begin
            writeln;
            write ('Enter Today''s Date (YYJJJ): ');
            TD:=Get_Real(0.0,99999.0);
            delete (PD82str5,211,15);
            insert (NumToString(TD),PD82str5,211);
          end;
      chr(13): done:=TRUE
  end;
  until done=TRUE;
  assign (outfile,'c:\tp\pd82in.fil');
  rewrite (outfile);
  writeln(outfile,PD82str1, PD82str2, PD82str3, PD82str4, PD82str5, PD82str6,
       PD82str7, PD82str8);
  close (outfile);
  clrscr;
end;


procedure InitPD86File;

var infile, outfile:text;

   PD82str1: string[24];
   PD82str2, PD82str3, PD82str4, PD82str5, PD82str6, PD82str7,
   PD82str8: string[255];

   PD86str1: string[24];
   PD86str2, PD86str3, PD86str4, PD86str5, PD86str6, PD86str7,
   PD86str8: string[255];
   PD86str9: string[60];

   C003,C001W:string[2];
   C001B,LASTIN,C001T1,C001T2,RPRIN,ONEWAY:char;
   FILLER:string[5];
   D046D:string[9];                    {NIIN}

   B011A,B073,FMLTCNT,FMLYEXP,FMLYGRS,FMLYMNM,FMLYSYSORD,FMLYSYSRO,
FMLYOPAST,FMLYPLT,FMLYRPRSRV,FMLYRTAT,FMLYRQSIZ,FSQPPR1,FSQPPR2,FSQPPR
3,FSQPPR4,FSQPPR5,FSQPPR6,FSQPPR7,FSQPPR8,FSQPPR9,FSQPPR10,FSQPPR11,FSQP
PR12,FSQPPR13,FSQPPR14,FSQPPR15,FSQPPR16,FSQPPR17,FSQPPR18,FSQPPR19,FSQP
PR20,FSQPPR21,FSQPPR22,FSQPPR23,FSQPPR24,FSQPPR25,FSQPPR26,FSQPPR27,FSQP
PR28,FSQPPR29,FSQPPR30,FSQPPR31,FSQPPR32,FWO,B023D,HRZNLNGTH,
```

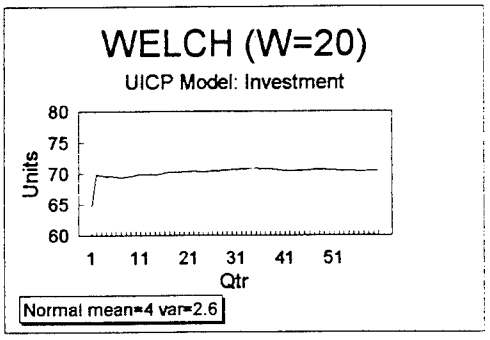
```
MEANNONZR,B061B,B019A,B019B,B019C,B021,B019,B021A,OPAST,PLTPPR,B012F,PPV,
   PPVO,BRLDCU,F009,B012E,RSV,SQPPR1,SQPPR2,SQPPR3,SQPPR4,SQPPR5,SQPPR6,
   SQPPR7,SQPPR8,SQPPR9,SQPPR10,SQPPR11,SQPPR12,SQPPR13,SQPPR14,SQPPR15,

SQPPR16,SQPPR17,SQPPR18,SQPPR19,SQPPR20,SQPPR21,SQPPR22,SQPPR23,SQPPR24
,
   SQPPR25, SQPPR26,SQPPR27,SQPPR28,SQPPR29,SQPPR30,SQPPR31,SQPPR32,
   SYSBO,SYSRCR,A023B,TRPR,TSDRS,B055,F007,ZOBS,EXPDEFRS,EXPDEFRSR,
   EXPDEFSDR,FEXPDEFRS,FEXPDEFSDR,PROJADDBO,PROJADDVRBL,PROJSMAVRBL,

PROJSSADDBO,PROJSSADD,PROJSSSMA,RQSHRTRND,RQSHRTYR,VLBUYS,VRBLHRSR,
   VRBLHRSQ,UNITSHRTP,UNITSSHRTR:real;



begin
  assign (infile,'c:\tp\pd82out.fil');
  reset (infile);
  read(infile,PD82str1, PD82str2, PD82str3, PD82str4, PD82str5, PD82str6,
      PD82str7, PD82str8);
  close (infile);
  C003:='1H';
  C001B:=' ';
  LASTIN:='Y';
  D046D:='000000000';                    {NIIN}
  C001T1:=' ';
  C001T2:=' ';
  C001W:='  ';
  RPRIN:='N';
  ONEWAY:='N';
  FILLER:='     ';
  strTemp:=copy(PD82str2,46,15);  B011A:=StringToReal(StrTemp);
  strTemp:=copy(PD82str3,31,15);  B073:=StringToReal(StrTemp);
  FMLTCNT:=0.0;FMLYEXP:=0.0;FMLYGRS:=0.0;FMLYMNM:=0.0;FMLYSYSORD:=0.0;

FMLYSYSRO:=0.0;FMLYOPAST:=0.0;FMLYPLT:=0.0;FMLYRPRSRV:=0.0;FMLYRTAT:=
0.0;
  FMLYRQSIZ:=0.0;FSQPPR1:=0.0;FSQPPR2:=0.0;FSQPPR3:=0.0;FSQPPR4:=0.0;
  FSQPPR5:=0.0;FSQPPR6:=0.0;FSQPPR7:=0.0;FSQPPR8:=0.0;FSQPPR9:=0.0;
  FSQPPR10:=0.0;FSQPPR11:=0.0;FSQPPR12:=0.0;FSQPPR13:=0.0;FSQPPR14:=0.0;
  FSQPPR15:=0.0;FSQPPR16:=0.0;FSQPPR17:=0.0;FSQPPR18:=0.0;FSQPPR19:=0.0;
  FSQPPR20:=0.0;FSQPPR21:=0.0;FSQPPR22:=0.0;FSQPPR23:=0.0;FSQPPR24:=0.0;
  FSQPPR25:=0.0;FSQPPR26:=0.0;FSQPPR27:=0.0;FSQPPR28:=0.0;FSQPPR29:=0.0;
  FSQPPR30:=0.0;FSQPPR31:=0.0;FSQPPR32:=0.0;FWO:=0.0;
  strTemp:=copy(PD82str2,121,15); B023D:=StringToReal(StrTemp);
  HRZNLNGTH:=0.0;MEANNONZR:=0.0;B061B:=0.0;
  strTemp:=copy(PD82str2,76,15);  B019A:=StringToReal(StrTemp);
  B019B:=0.0;B019C:=0.0;
  strTemp:=copy(PD82str7,226,15); B021:=StringToReal(StrTemp);
```

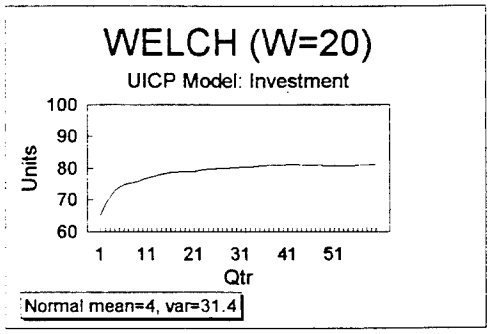
```
  strTemp:=copy(PD82str7,196,15); B019:=StringToReal(StrTemp);
  B021A:=0.0;OPAST:=0.0;PLTPPR:=0.0;B012F:=0.0;
  strTemp:=copy(PD82str5,91,15);  PPV:=StringToReal(StrTemp);
  PPVO:=0.0;
  strTemp:=copy(PD82str7,121,15); BRLDCU:=StringToReal(StrTemp);
  F009:=0.0;B012E:=0.0;
  RSV:=0.0;
  SQPPR1:=0.0;SQPPR2:=0.0;

SQPPR3:=0.0;SQPPR4:=0.0;SQPPR5:=0.0;SQPPR6:=0.0;SQPPR7:=0.0;SQPPR8:=0.0;
  SQPPR9:=0.0;SQPPR10:=0.0;SQPPR11:=0.0;SQPPR12:=0.0;SQPPR13:=0.0;
  SQPPR14:=0.0;SQPPR15:=0.0;SQPPR16:=0.0;SQPPR17:=0.0;SQPPR18:=0.0;
  SQPPR19:=0.0;SQPPR20:=0.0;SQPPR21:=0.0;SQPPR22:=0.0;SQPPR23:=0.0;
  SQPPR24:=0.0;SQPPR25:=0.0; SQPPR26:=0.0;SQPPR27:=0.0;SQPPR28:=0.0;
  SQPPR29:=0.0;SQPPR30:=0.0;SQPPR31:=0.0;SQPPR32:=0.0;
  SYSBO:=0.0;SYSRCR:=0.0;
  strTemp:=copy(PD82str2,1,15);   A023B:=StringToReal(StrTemp);
  strTemp:=copy(PD82str5,226,15); TRPR:=StringToReal(StrTemp);
  strTemp:=copy(PD82str5,226,15); TSDRS:=StringToReal(StrTemp);
  strTemp:=copy(PD82str2,181,15); B055:=StringToReal(StrTemp);
  F007:=0.0;ZOBS:=0.0;

EXPDEFRS:=0.0;EXPDEFRSR:=0.0;EXPDEFSDR:=0.0;FEXPDEFRS:=0.0;FEXPDEFSDR:=0
.0;
  PROJADDBO:=0.0;PROJADDVRBL:=0.0;PROJSMAVRBL:=0.0;PROJSSADDBO:=0.0;

PROJSSADD:=0.0;PROJSSSMA:=0.0;RQSHRTRND:=0.0;RQSHRTYR:=0.0;VLBUYS:=0.
0;
  VRBLHRSR:=0.0;VRBLHRSQ:=0.0;UNITSHRTP:=0.0;UNITSSHRTR:=0.0;

  {create PD86 input file}

  PD86str1:=C003+ C001B+ LASTIN+ D046D+ C001T1+ C001T2+ C001W+
RPRIN+ ONEWAY+
        FILLER;
  PD86str2:=NumToString(B011A)+NumToString(B073)+NumToString(FMLTCNT)+
        NumToString(FMLYEXP)+NumToString(FMLYGRS)+NumToString(FMLYMNM)+
        NumToString(FMLYSYSORD)+NumToString(FMLYSYSRO)+
        NumToString(FMLYOPAST)+NumToString(FMLYPLT)+
        NumToString(FMLYRPRSRV)+NumToString(FMLYRTAT)+
        NumToString(FMLYRQSIZ)+NumToString(FSQPPR1)+
        NumToString(FSQPPR2)+NumToString(FSQPPR3)+NumToString(FSQPPR4);

PD86str3:=NumToString(FSQPPR5)+NumToString(FSQPPR6)+NumToString(FSQPPR7)+
        NumToString(FSQPPR8)+NumToString(FSQPPR9)+NumToString(FSQPPR10)+
        NumToString(FSQPPR11)+NumToString(FSQPPR12)+
        NumToString(FSQPPR13)+NumToString(FSQPPR14)+
        NumToString(FSQPPR15)+NumToString(FSQPPR16)+
        NumToString(FSQPPR17)+NumToString(FSQPPR18)+
        NumToString(FSQPPR19)+NumToString(FSQPPR20)+
```

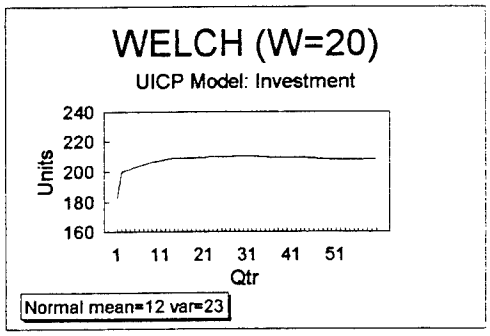
```
        NumToString(FSQPPR21);
    PD86str4:= NumToString(FSQPPR22) + NumToString(FSQPPR23) +
        NumToString(FSQPPR24) + NumToString(FSQPPR25) +
        NumToString(FSQPPR26) + NumToString(FSQPPR27) +
        NumToString(FSQPPR28) + NumToString(FSQPPR29) +
        NumToString(FSQPPR30) + NumToString(FSQPPR31) +
        NumToString(FSQPPR32) + NumToString(FWO) +
        NumToString(B023D) + NumToString(HRZNLNGTH) +
        NumToString(MEANNONZR) + NumToString(B061B) + NumToString(B019A);
    PD86str5:= NumToString(B019B) + NumToString(B019C) + NumToString(B021) +
        NumToString(B019) + NumToString(B021A) + NumToString(OPAST) +
        NumToString(PLTPPR) + NumToString(B012F) + NumToString(PPV) +
        NumToString(PPVO) + NumToString(BRLDCU) + NumToString(F009) +
        NumToString(B012E) + NumToString(RSV) + NumToString(SQPPR1) +
        NumToString(SQPPR2) + NumToString(SQPPR3);
    PD86str6:= NumToString(SQPPR4) + NumToString(SQPPR5) + NumToString(SQPPR6) +
        NumToString(SQPPR7) + NumToString(SQPPR8) + NumToString(SQPPR9) +
        NumToString(SQPPR10) + NumToString(SQPPR11) + NumToString(SQPPR12) +
        NumToString(SQPPR13) + NumToString(SQPPR14) + NumToString(SQPPR15) +
        NumToString(SQPPR16) + NumToString(SQPPR17) + NumToString(SQPPR18) +
        NumToString(SQPPR19) + NumToString(SQPPR20);
    PD86str7:= NumToString(SQPPR21) + NumToString(SQPPR22) +
        NumToString(SQPPR23) + NumToString(SQPPR24) + NumToString(SQPPR25) +
        NumToString(SQPPR26) + NumToString(SQPPR27) + NumToString(SQPPR28) +
        NumToString(SQPPR29) + NumToString(SQPPR30) + NumToString(SQPPR31) +
        NumToString(SQPPR32) + NumToString(SYSBO) + NumToString(SYSRCR) +
        NumToString(A023B) + NumToString(TRPR) + NumToString(TSDRS);
    PD86str8:= NumToString(B055) + NumToString(F007) +
        NumToString(ZOBS) + NumToString(EXPDEFRS) + NumToString(EXPDEFRSR) +
        NumToString(EXPDEFSDR) + NumToString(FEXPDEFRS) +
        NumToString(FEXPDEFSDR) + NumToString(PROJADDBO) +
        NumToString(PROJADDVRBL) + NumToString(PROJSMAVRBL) +
        NumToString(PROJSSADDBO) + NumToString(PROJSSADD) +
        NumToString(PROJSSSMA) + NumToString(RQSHRTRND) +
        NumToString(RQSHRTYR) + NumToString(VLBUYS);
    PD86str9:= NumToString(VRBLHRSR) + NumToString(VRBLHRSQ) +
        NumToString(UNITSHRTP) + NumToString(UNITSSHRTR);

    assign (outfile,'c:\tp\pd86in.fil');
    rewrite (outfile);
    writeln(outfile,PD86str1, PD86str2, PD86str3, PD86str4, PD86str5, PD86str6,
        PD86str7, PD86str8, PD86str9);
close (outfile);
end;

end. {unit pdunit}
```

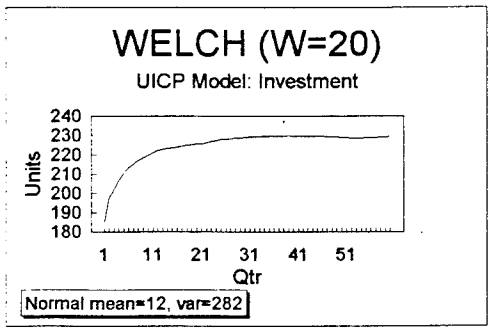
APPENDIX B. WELCH GRAPHS

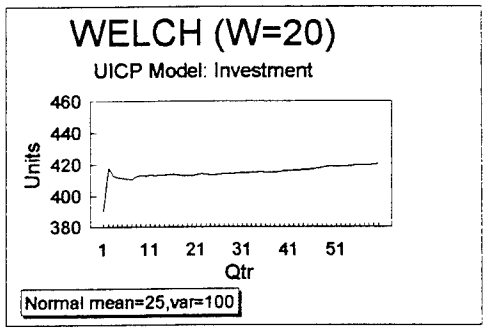


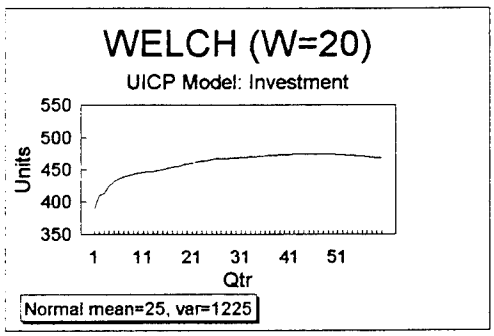


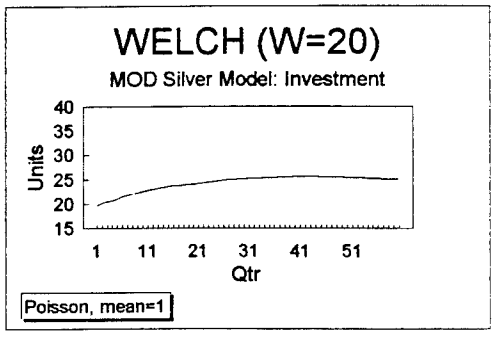


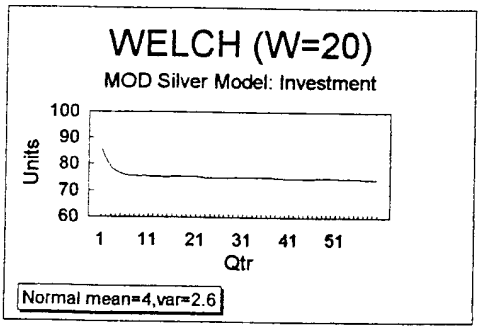


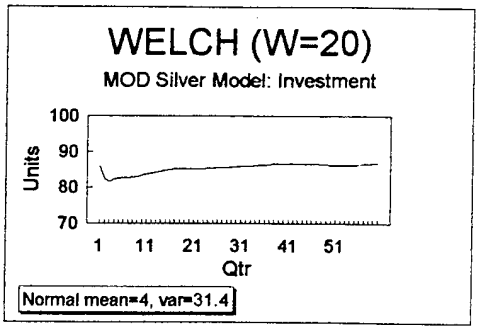


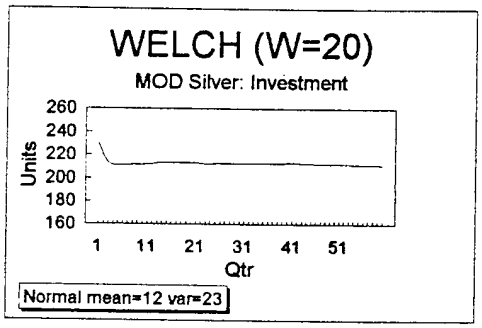


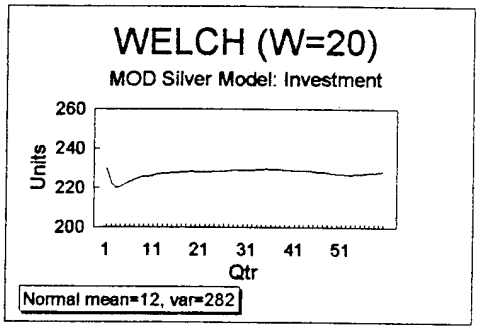


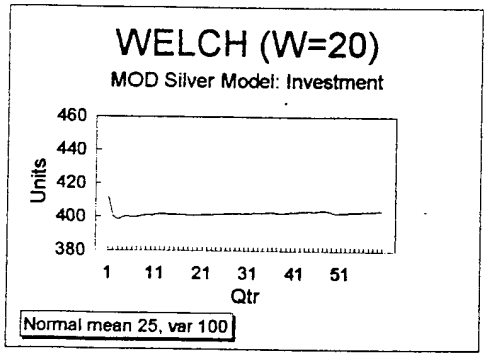


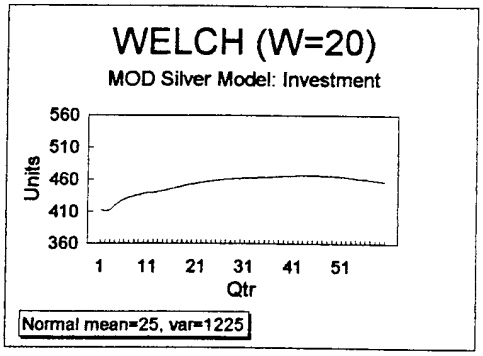

WELCH (W=20)
MOD Silver Model: Investment
Units
100
90
80
70
60
1 11 21 31 41 51
Qtr
Normal mean=4,var=2.6


WELCH (W=20)
MOD Silver Model: Investment
Units
100
90
80
70
1 11 21 31 41 51
Qtr
Normal mean=4, var=31.4


WELCH (W=20)
MOD Silver: Investment
Units
260
240
220
200
180
160
1 11 21 31 41 51
Qtr
Normal mean=12 var=23


WELCH (W=20)
MOD Silver Model: Investment
Units
260
240
220
200
1 11 21 31 41 51
Qtr
Normal mean=12, var=282


WELCH (W=20)
MOD Silver Model: Investment
Units
460
440
420
400
380
1 11 21 31 41 51
Qtr
Normal mean 25, var 100


WELCH (W=20)
MOD Silver Model: Investment
Units
560
510
460
410
360
1 11 21 31 41 51
Qtr
Normal mean=25, var=1225

APPENDIX C. SCENARIO LISTING

This appendix contains a listing of the scenarios (experiments) used in the performance testing of the modified Silver model. Each simulation experiment consisted of 500 replications. A description of the list headings follows:

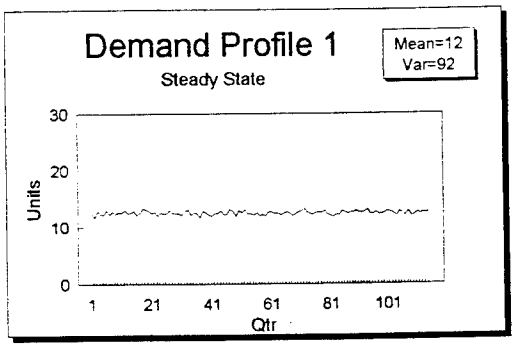
Exp #	Experiment number
Qtrs	Total number of quarters
Strt	Starting quarter for collecting steady state statistics
End	Ending quarter for collecting steady state statistics
Dist	Distribution used to generate demand observations
Mean	Mean used to generate demand observation (initial mean)
Var	Variance used to generate demand observations (initial variance)
T/S Qtr	Trend Starting quarter
T/E Qtr	Trend Ending quarter
Coeff	Trend coefficient value
Pwr	Trend power value
PLT	Mean procurement lead time (F suffix implies fixed lead time)
U/P	Item unit price or cost
Risk	Risk value used in the modified Silver model
Profile	Demand profile (see Appendix D)

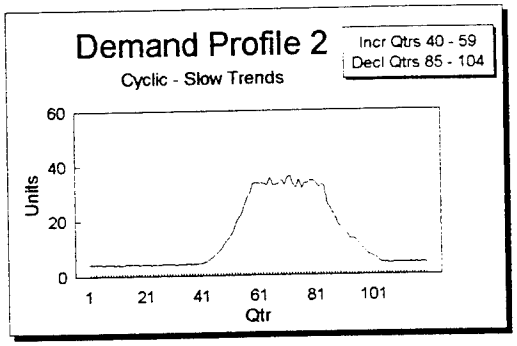
Exp #	Qtrs	Strt	End	Dist	Mean	Var	T/S Qtr	T/E Qtr	Coeff	Pwr	PLT	U/P	Risk	Profile
1	115	26	105	P	0.25						8	5000	0.3	1
2	115	26	105	P	1						8	1000	0.21	1
3	115	26	105	N	4	2.6					8	250	0.12	1
4	115	26	105	N	4	10.2					8	250	0.11	1
5	115	26	105	N	4	31.4					8	250	0.11	1
6	115	26	105	N	12	23					8	100	0.12	1
7	115	26	105	N	12	92					8	100	0.1	1
8	115	26	105	N	12	282					8	100	0.1	1
9	115	26	105	N	25	100					8	50	0.1	1
10	115	26	105	N	25	400					8	50	0.1	1
11	115	26	105	N	25	1225					8	50	0.1	1
12	115	26	105	P	0.25						8	20000	0.31	1
13	115	26	105	P	1						8	5000	0.31	1
14	115	26	105	N	4	2.6					8	1250	0.32	1
15	115	26	105	N	4	10.2					8	1250	0.34	1
16	115	26	105	N	4	31.4					8	1250	0.34	1
17	115	26	105	N	12	23					8	450	0.17	1
18	115	26	105	N	12	92					8	450	0.16	1
19	115	26	105	N	12	282					8	450	0.15	1
20	115	26	105	N	25	100					8	200	0.1	1
21	115	26	105	N	25	400					8	200	0.1	1
22	115	26	105	N	25	1225					8	200	0.11	1
23	115	26	105	P	1						4	1000	0.29	1
24	115	26	105	N	4	2.6					4	250	0.15	1
25	115	26	105	N	4	31.4					4	250	0.13	1
26	115	26	105	N	12	23					4	100	0.14	1
27	115	26	105	N	12	282					4	100	0.11	1
28	115	26	105	P	1						8F	1000	0.16	1
29	115	26	105	N	4	2.6					8F	250	0.15	1
30	115	26	105	N	4	31.4					8F	250	0.13	1
31	115	26	105	N	12	23					8F	100	0.15	1
32	115	26	105	N	12	282					8F	100	0.1	1
33	120	26	110	P	0.25		40	59	0.0175	2	8	1500	0.3	2
							85	104	-0.195	0.5				
34	120	26	110	N	4	2.6	40	59	0.0175	2	8	125	0.1	2
							85	104	-0.195	0.5				
35	120	26	110	N	4	10.2	40	59	0.0175	2	8	125	0.1	2
							85	104	-0.195	0.5				
36	120	26	110	N	4	31.4	40	59	0.0175	2	8	125	0.1	2
							85	104	-0.195	0.5				

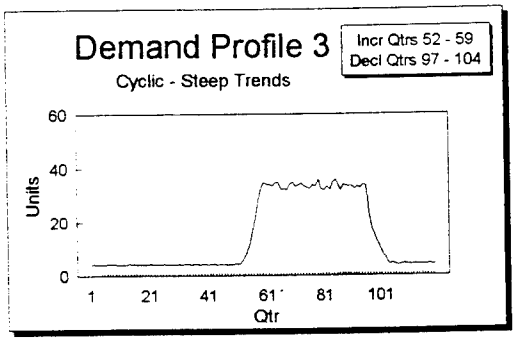
Exp #	Qtrs	Strt	End	Dist	Mean	Var	T/S Qtr	T/E Qtr	Coeff	Pwr	PLT	U/P	Risk	Profile
37	120	26	110	P	0.25		52	59	0.11	2	8	1500	0.3	3
							97	104	-0.31	0.5				
38	120	26	110	N	4	2.6	52	59	0.11	2	8	125	0.1	3
							97	104	-0.31	0.5				
39	120	26	110	N	4	10.2	52	59	0.11	2	8	125	0.1	3
							97	104	-0.31	0.5				
40	120	26	110	N	4	31.4	52	59	0.11	2	8	125	0.1	3
							97	104	-0.31	0.5				
41	75	26	65	P	1		40	59	-0.17	0.5	8	1000	0.21	4
42	75	26	65	N	25	100	40	59	-0.17	0.5	8	100	0.1	4
43	75	26	65	N	25	400	40	59	-0.17	0.5	8	100	0.1	4
44	75	26	65	N	25	1225	40	59	-0.17	0.5	8	100	0.1	4
45	75	26	65	P	1		52	59	-0.268	0.5	8	1000	0.21	5
46	75	26	65	N	25	100	52	59	-0.268	0.5	8	100	0.1	5
47	75	26	65	N	25	400	52	59	-0.268	0.5	8	100	0.1	5
48	75	26	65	N	25	1225	52	59	-0.268	0.5	8	100	0.1	5
49	75	26	65	P	1		40	59	-0.038	1	8	1000	0.21	6
50	75	26	65	N	25	100	40	59	-0.038	1	8	100	0.1	6
51	75	26	65	N	25	400	40	59	-0.038	1	8	100	0.1	6
52	75	26	65	N	25	1225	40	59	-0.038	1	8	100	0.1	6
53	75	26	65	P	1		52	59	-0.095	1	8	1000	0.21	7
54	75	26	65	N	25	100	52	59	-0.095	1	8	100	0.1	7
55	75	26	65	N	25	400	52	59	-0.095	1	8	100	0.1	7
56	75	26	65	N	25	1225	52	59	-0.095	1	8	100	0.1	7
57	75	26	65	P	1		40	59	-0.0085	1.5	8	1000	0.21	8
58	75	26	65	N	25	100	40	59	-0.0085	1.5	8	100	0.1	8
59	75	26	65	N	25	400	40	59	-0.0085	1.5	8	100	0.1	8
60	75	26	65	N	25	1225	40	59	-0.0085	1.5	8	100	0.1	8
61	75	26	65	P	1		52	59	-0.0335	1.5	8	1000	0.21	9
62	75	26	65	N	25	100	52	59	-0.0335	1.5	8	100	0.1	9
63	75	26	65	N	25	400	52	59	-0.0335	1.5	8	100	0.1	9
64	75	26	65	N	25	1225	52	59	-0.0335	1.5	8	100	0.1	9
65	75	26	65	P	1		40	59	-0.092	0.8	8	1000	0.21	10
66	75	26	65	N	25	100	40	59	-0.092	0.8	8	100	0.1	10
67	75	26	65	N	25	400	40	59	-0.092	0.8	8	100	0.1	10
68	75	26	65	N	25	1225	40	59	-0.092	0.8	8	100	0.1	10
69	75	26	65	P	1		52	59	-0.19	0.8	8	1000	0.21	11
70	75	26	65	N	25	100	52	59	-0.19	0.8	8	100	0.1	11
71	75	26	65	N	25	400	52	59	-0.19	0.8	8	100	0.1	11
72	75	26	65	N	25	1225	52	59	-0.19	0.8	8	100	0.1	11

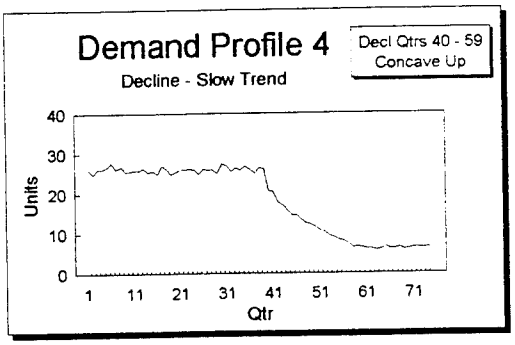
Exp #	Qtrs	Strt	End	Dist	Mean	Var	T/S Qtr	T/E Qtr	Coeff	Pwr	PLT	U/P	Risk	Profile
73	75	26	65	P	1		40		0.75		8	1000	0.21	12
							44		0.75					
							48		0.75					
							52		0.75					
							56		0.75					
74	75	26	65	N	25	100	40		0.75		8	100	0.1	12
							44		0.75					
							48		0.75					
							52		0.75					
							56		0.75					
75	75	26	65	N	25	400	40		0.75		8	100	0.1	12
							44		0.75					
							48		0.75					
							52		0.75					
							56		0.75					
76	75	26	65	N	25	1225	40		0.75		8	100	0.1	12
							44		0.75					
							48		0.75					
							52		0.75					
							56		0.75					
77	100	26	90	P	0.25		40	59	0.0175	2	8	1000	0.25	13
78	100	26	90	N	4	2.6	40	59	0.0175	2	8	100	0.1	13
79	100	26	90	N	4	10.2	40	59	0.0175	2	8	100	0.1	13
80	100	26	90	N	4	31.4	40	59	0.0175	2	8	100	0.1	13
81	100	26	90	P	0.25		52	59	0.11	2	8	1000	0.25	14
82	100	26	90	N	4	2.6	52	59	0.11	2	8	100	0.1	14
83	100	26	90	N	4	10.2	52	59	0.11	2	8	100	0.1	14
84	100	26	90	N	4	31.4	52	59	0.11	2	8	100	0.1	14
85	100	26	90	P	0.25		40	59	1.565	0.5	8	1000	0.25	15
86	100	26	90	N	4	2.6	40	59	1.565	0.5	8	100	0.1	15
87	100	26	90	N	4	10.2	40	59	1.565	0.5	8	100	0.1	15
88	100	26	90	N	4	31.4	40	59	1.565	0.5	8	100	0.1	15
89	100	26	90	P	0.25		52	59	2.48	0.5	8	1000	0.25	16
90	100	26	90	N	4	2.6	52	59	2.48	0.5	8	100	0.1	16
91	100	26	90	N	4	10.2	52	59	2.48	0.5	8	100	0.1	16
92	100	26	90	N	4	31.4	52	59	2.48	0.5	8	100	0.1	16

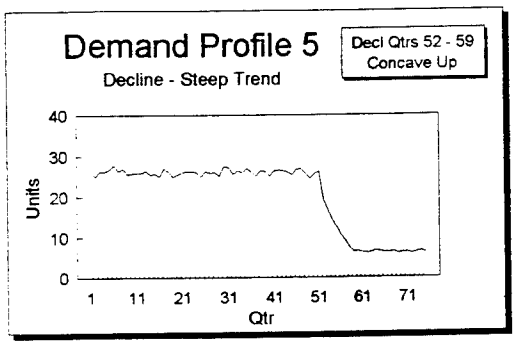
APPENDIX D. DEMAND PROFILE GRAPHS

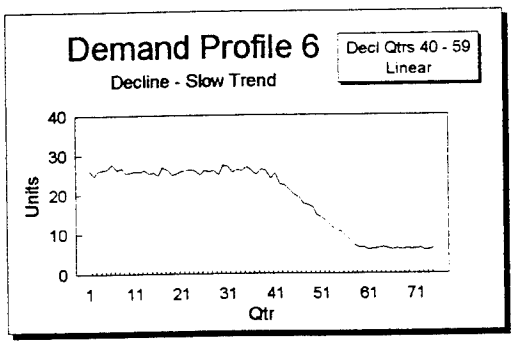


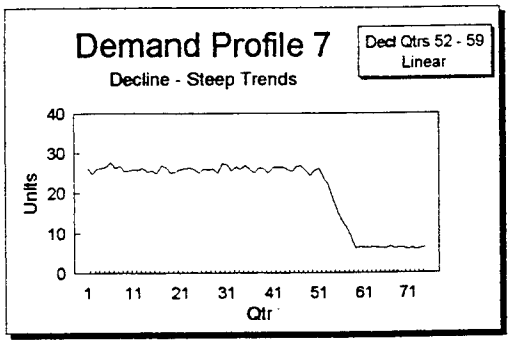


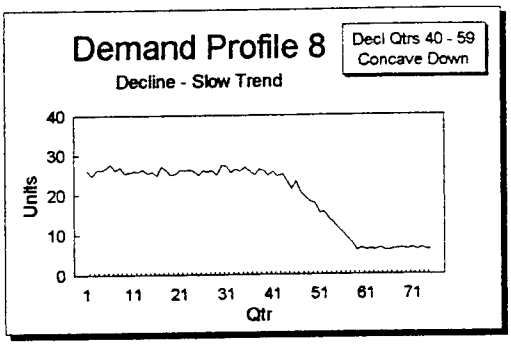


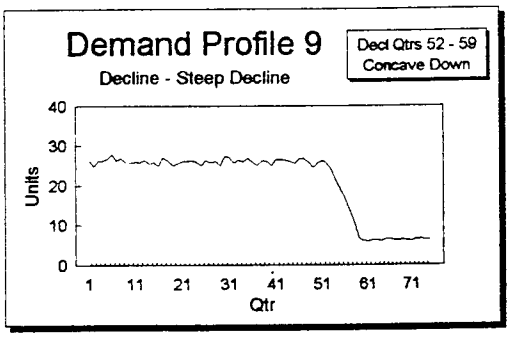


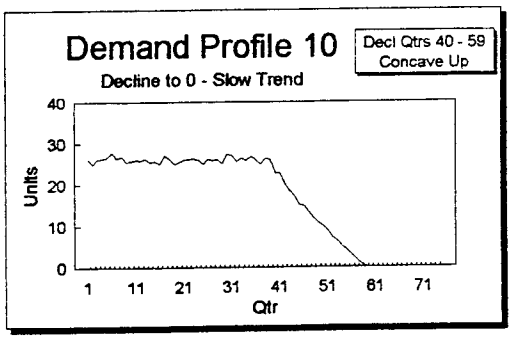


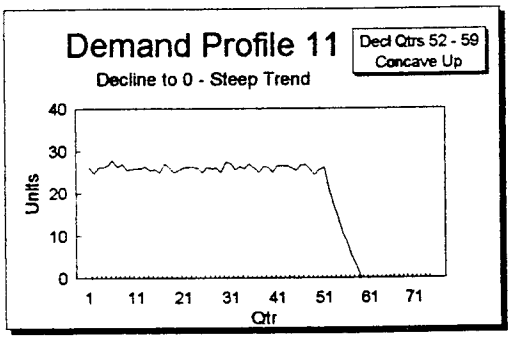


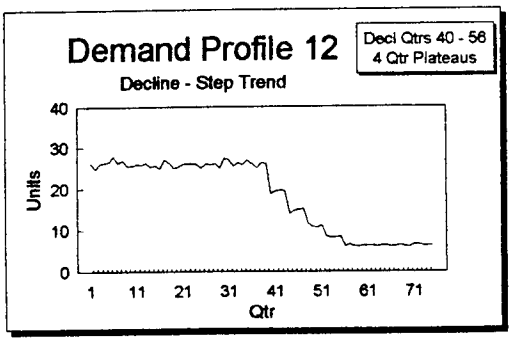

Demand Profile 7
Decl Qtrs 52 - 59
Linear
Decline - Steep Trends
Units
Qtr
0
10
20
30
40
1
11
21
31
41
51
61
71

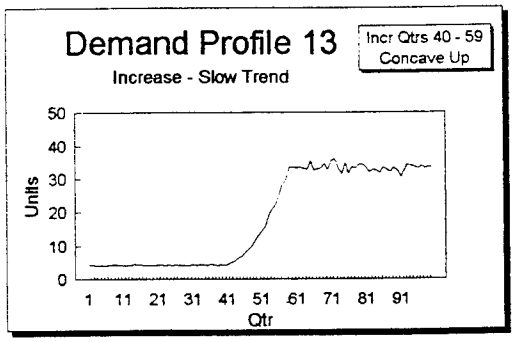

Demand Profile 8
Decl Qtrs 40 - 59
Concave Down
Decline - Slow Trend
Units
Qtr
0
10
20
30
40
1
11
21
31
41
51
61
71

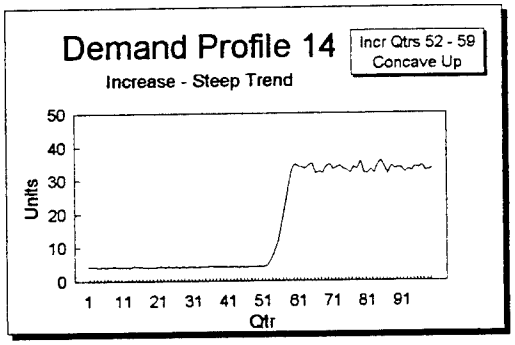

Demand Profile 9
Decl Qtrs 52 - 59
Concave Down
Decline - Steep Decline
Units
Qtr
0
10
20
30
40
1
11
21
31
41
51
61
71

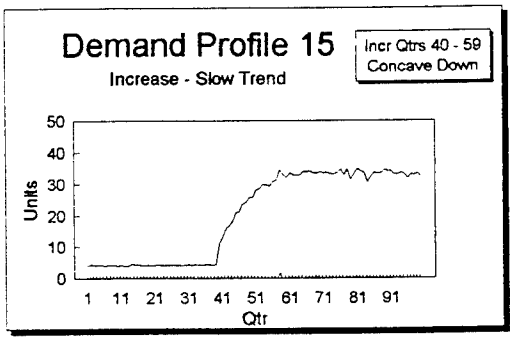

Demand Profile 10
Decl Qtrs 40 - 59
Concave Up
Decline to 0 - Slow Trend
Units
Qtr
0
10
20
30
40
1
11
21
31
41
51
61
71

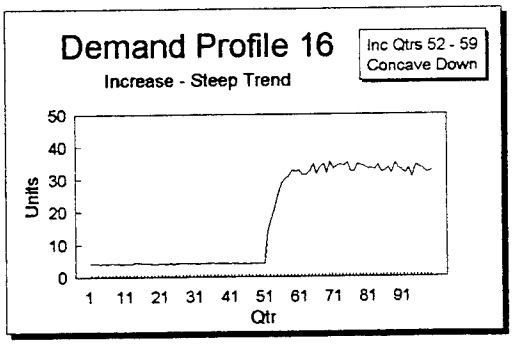

Demand Profile 11
Decl Qtrs 52 - 59
Concave Up
Decline to 0 - Steep Trend
Units
Qtr
0
10
20
30
40
1
11
21
31
41
51
61
71


Demand Profile 12
Decl Qtrs 40 - 56
4 Qtr Plateaus
Decline - Step Trend
Units
Qtr
0
10
20
30
40
1
11
21
31
41
51
61
71


Demand Profile 13
Incr Qtrs 40 - 59
Concave Up
Increase - Slow Trend
Units
50
40
30
20
10
0
1 11 21 31 41 51 61 71 81 91
Qtr


Demand Profile 14
Incr Qtrs 52 - 59
Concave Up
Increase - Steep Trend
Units
50
40
30
20
10
0
1 11 21 31 41 51 61 71 81 91
Qtr


Demand Profile 15
Incr Qtrs 40 - 59
Concave Down
Increase - Slow Trend
Units
50
40
30
20
10
0
1 11 21 31 41 51 61 71 81 91
Qtr


Demand Profile 16
Inc Qtrs 52 - 59
Concave Down
Increase - Steep Trend
Units
50
40
30
20
10
0
1 11 21 31 41 51 61 71 81 91
Qtr

APPENDIX E. EXPERIMENTAL RESULTS

This appendix contains a summary of the data obtained during the simulations conducted during this thesis. All statistics are collected over the specified steady state collection period (see Appendix C). Also included is the analysis data used in comparing the modified Silver and UICP models. A description of the table entries follows:

Exp# Experiment number. This number corresponds to the experiment number listed in Appendix C. When suffixed with an F it indicates that "fixed" forecasting was used for the modified Silver model.

Model SILVER - mean measure for modified Silver model
UICP - mean measure for UICP model
Mean Diff - difference in mean values (SILVER-UICP)
p-value - based on paired t-test on difference of the means (sample size=500)

ACWTBO Average customer wait time for backordered requisitions in days (cumulative measure)

ACWT Average customer wait time in days (cumulative measure)

SMA Supply material availability (cumulative measure)

Invest Average quarterly investment level in units

Total Cost Total cost in dollars (over the experimental time interval)

Excess Ending amount of excess inventory in units

Orders Total number of orders

The following is a list of tables found in this appendix:

Table E-1: Contains results for the stationary mean demand scenarios.

Table E-2: Contains results for the cyclic mean demand scenarios.

Table E-3: Contains results for the declining mean demand scenarios.

Table E-4: Contains results for the increasing mean demand scenarios.

Table E-5: Contains results for the cyclic mean demand scenarios using a "fixed" or stationary forecast assumption with the modified Silver model.

Table E-6: Contains results for the declining mean demand scenarios using a "fixed" or stationary forecast assumption with the modified Silver model.

Table E-7: Contains results for the increasing mean demand scenarios using a "fixed" or stationary forecast assumption with the modified Silver model.

TABLE E-1.

STATIONARY MEAN DEMAND								
Exp#	**Model**	**ACWTBO**	**ACWT**	**SMA**	**Invest**	**Total Cost**	**Excess**	**Orders**
1	SILVER	118.03	13.66	0.935	6.54	243,819.51	3.19	7.83
	UICP	121.48	16.03	0.921	6.64	246,711.33	3.36	8.51
	Mean Diff	-3.45	-2.37	0.014	-0.10	-2,891.83	-0.17	
	p-value	0.57	0.00	0.00	0.00	0.00	0.02	
2	SILVER	84.50	6.53	0.963	19.96	164,251.35	5.68	10.19
	UICP	67.46	4.39	0.971	19.77	162,800.71	5.68	10.07
	Mean Diff	17.04	2.15	-0.008	0.19	1,450.64	0.01	
	p-value	0.00	0.00	0.00	0.00	0.00	0.96	
3	SILVER	29.06	1.00	0.990	68.55	150,651.52	9.26	13.60
	UICP	33.72	1.08	0.989	69.80	151,631.12	10.04	11.89
	Mean Diff	-4.66	-0.08	0.001	-1.25	-979.60	-0.78	
	p-value	0.02	0.44	0.34	0.00	0.00	0.22	
4	SILVER	54.33	3.65	0.977	81.14	170,971.83	20.40	11.67
	UICP	65.33	4.41	0.971	80.96	170,267.39	19.67	10.22
	Mean Diff	-11.00	-0.75	0.006	0.18	704.44	0.73	
	p-value	0.00	0.00	0.00	0.23	0.04	0.32	
5	SILVER	88.77	7.25	0.962	100.43	208,004.93	31.13	11.04
	UICP	96.83	8.60	0.955	102.43	210,999.50	34.78	9.36
	Mean Diff	-8.06	-1.36	0.007	-1.99	-2,994.57	-3.65	
	p-value	0.06	0.00	0.00	0.00	0.00	0.00	
6	SILVER	21.53	0.72	0.993	203.56	180,178.57	24.03	15.42
	UICP	33.04	1.14	0.989	206.30	180,548.91	28.94	12.07
	Mean Diff	-11.50	-0.42	0.004	-2.74	-370.33	-4.91	
	p-value	0.00	0.00	0.00	0.00	0.31	0.01	
7	SILVER	48.68	2.58	0.981	233.72	202,052.43	48.85	13.47
	UICP	67.72	4.62	0.970	234.35	205,784.45	47.29	10.62
	Mean Diff	-19.04	-2.04	0.011	-0.63	-3,732.03	1.15	
	p-value	0.00	0.00	0.00	0.14	0.00	0.63	

TABLE E-1. (CONTINUED)

STATIONARY MEAN DEMAND								
Exp#	**Model**	**ACWTBO**	**ACWT**	**SMA**	**Invest**	**Total Cost**	**Excess**	**Orders**
8	SILVER	78.87	6.04	0.965	288.32	251,289.97	87.28	12.38
	UICP	101.06	8.42	0.954	294.18	259,396.05	93.49	9.48
	Mean Diff	-22.19	-2.37	0.011	-5.86	-8,106.07	-6.21	
	p-value	0.00	0.00	0.00	0.00	0.00	0.04	
9	SILVER	24.68	0.67	0.993	416.48	186,182.59	52.11	15.94
	UICP	37.30	1.33	0.987	426.21	189,617.22	60.31	11.95
	Mean Diff	-12.62	-0.66	0.006	-9.73	-3,434.63	-8.20	
	p-value	0.00	0.00	0.00	0.00	0.00	0.01	
10	SILVER	52.21	3.25	0.977	486.56	221,305.92	106.35	13.54
	UICP	82.03	5.51	0.964	493.97	233,251.97	120.03	10.21
	Mean Diff	-29.82	-2.27	0.013	-7.41	-11,946.05	-13.68	
	p-value	0.00	0.00	0.00	0.00	0.00	0.01	
11	SILVER	84.28	7.00	0.960	595.19	285,641.02	159.51	12.60
	UICP	104.60	9.89	0.950	610.22	305,649.74	172.04	9.33
	Mean Diff	-20.32	-2.89	0.010	-15.03	-20,008.71	-12.53	
	p-value	0.00	0.00	0.00	0.00	0.00	0.05	
12	SILVER	121.26	15.70	0.917	5.61	856,925.35	2.27	13.85
	UICP	128.83	18.94	0.897	5.74	867,918.12	2.52	13.91
	Mean Diff	-7.57	-3.24	0.021	-0.13	-10,992.76	-0.25	
	p-value	0.09	0.00	0.00	0.00	0.00	0.00	
13	SILVER	111.18	11.93	0.922	15.75	679,467.98	2.58	22.99
	UICP	110.65	11.00	0.929	15.96	676,481.49	2.77	15.79
	Mean Diff	0.54	0.93	-0.007	-0.21	2,986.50	-0.19	
	p-value	0.90	0.11	0.01	0.00	0.01	0.10	
14	SILVER	80.73	7.36	0.930	48.30	591,463.60	0.55	37.76
	UICP	117.53	13.67	0.896	49.64	591,463.60	0.55	18.50
	Mean Diff	-36.80	-6.31	0.034	-1.34	1,278.53	-0.71	
	p-value	0.00	0.00	0.00	0.00	0.09	0.00	

TABLE E-1. (CONTINUED)

STATIONARY MEAN DEMAND								
Exp#	**Model**	**ACWTBO**	**ACWT**	**SMA**	**Invest**	**Total Cost**	**Excess**	**Orders**
15	SILVER	132.65	20.29	0.873	54.05	647,714.09	3.37	31.25
	UICP	142.44	22.05	0.867	55.33	644,085.16	4.14	17.22
	Mean Diff	-9.79	-1.76	0.006	-1.28	3,628.93	-0.77	
	p-value	0.01	0.04	0.07	0.00	0.00	0.02	
16	SILVER	168.06	32.97	0.829	65.60	775,840.79	7.82	29.07
	UICP	162.97	28.39	0.849	68.10	775,960.49	9.71	16.91
	Mean Diff	5.09	4.58	-0.019	-2.49	-119.71	-1.89	
	p-value	0.24	0.00	0.00	0.00	0.94	0.00	
17	SILVER	34.14	1.41	0.984	165.25	689,439.35	5.76	42.44
	UICP	57.77	2.93	0.972	173.78	694,617.56	9.07	18.65
	Mean Diff	-23.63	-1.52	0.011	-8.53	-5,178.21	-3.31	
	p-value	0.00	0.00	0.00	0.00	0.00	0.00	
18	SILVER	68.03	5.37	0.957	188.40	757,643.65	18.10	33.57
	UICP	91.92	7.50	0.948	193.57	761,568.88	21.30	16.73
	Mean Diff	-23.89	-2.13	0.009	-5.17	-3,925.24	-3.20	
	p-value	0.00	0.00	0.00	0.00	0.01	0.02	
19	SILVER	110.28	12.62	0.923	237.38	936,052.79	45.62	28.71
	UICP	121.23	13.25	0.924	240.29	932,062.39	48.60	15.80
	Mean Diff	-10.95	-0.63	-0.001	-2.91	3,990.41	-2.98	
	p-value	0.01	0.41	0.76	0.00	0.13	0.06	
20	SILVER	11.83	0.39	0.995	377.33	671,529.57	21.37	38.46
	UICP	36.27	1.19	0.987	393.67	675,002.46	34.07	17.58
	Mean Diff	-24.44	-0.79	0.008	-16.34	-3,472.89	-12.70	
	p-value	0.00	0.00	0.00	0.00	0.00	0.00	
21	SILVER	51.50	3.25	0.975	442.03	762,426.80	72.54	30.13
	UICP	78.97	5.72	0.962	445.61	765,254.51	76.46	15.19
	Mean Diff	-27.47	-2.47	0.013	-3.58	-2,827.71	-3.92	
	p-value	0.00	0.00	0.00	0.00	0.25	0.25	

TABLE E-1. (CONTINUED)

STATIONARY MEAN DEMAND								
Exp#	**Model**	**ACWTBO**	**ACWT**	**SMA**	**Invest**	**Total Cost**	**Excess**	**Orders**
22	SILVER	87.83	8.55	0.947	537.67	933,637.94	118.90	26.13
	UICP	104.95	10.34	0.941	546.04	940,914.74	123.74	14.29
	Mean Diff	-17.12	-1.79	0.006	-8.38	-7,276.80	-4.84	
	p-value	0.00	0.00	0.01	0.00	0.06	0.26	
23	SILVER	93.44	6.99	0.943	11.90	138,388.02	2.13	11.87
	UICP	84.52	5.45	0.953	12.08	138,592.40	2.14	12.03
	Mean Diff	8.92	1.54	-0.010	-0.18	-204.38	-0.01	
	p-value	0.01	0.00	0.00	0.00	0.42	0.91	
24	SILVER	37.36	1.10	0.987	44.64	135,753.90	2.91	24.23
	UICP	42.57	1.33	0.985	45.58	135,534.91	4.06	12.60
	Mean Diff	-5.21	-0.23	0.002	-0.94	218.98	-1.15	
	p-value	0.01	0.02	0.01	0.00	0.20	0.00	
25	SILVER	77.08	5.20	0.958	62.73	176,492.70	14.10	13.14
	UICP	83.07	6.39	0.950	64.82	178,627.70	15.87	11.26
	Mean Diff	-6.00	-1.19	0.008	-2.09	-2,135.00	-1.77	
	p-value	0.05	0.00	0.00	0.00	0.00	0.00	
26	SILVER	30.10	0.87	0.989	131.34	162,101.05	6.64	16.11
	UICP	40.59	1.33	0.984	134.74	161,885.00	10.30	12.71
	Mean Diff	-10.49	-0.46	0.005	-3.40	216.04	-3.66	
	p-value	0.00	0.00	0.00	0.00	0.50	0.00	
27	SILVER	71.60	4.36	0.963	181.18	213,038.80	36.33	14.76
	UICP	86.07	6.43	0.947	185.72	218,957.21	41.26	11.47
	Mean Diff	-14.48	-2.07	0.016	-4.54	-5,918.41	-4.92	
	p-value	0.00	0.00	0.00	0.00	0.00	0.00	
28	SILVER	70.15	4.72	0.966	19.53	161,368.05	5.29	10.08
	UICP	51.44	2.60	0.978	19.70	161,997.69	5.25	10.07
	Mean Diff	18.71	2.12	-0.012	-0.17	-629.63	0.04	
	p-value	0.00	0.00	0.00	0.01	0.05	0.75	

TABLE E-1. (CONTINUED)

STATIONARY MEAN DEMAND								
Exp#	**Model**	**ACWTBO**	**ACWT**	**SMA**	**Invest**	**Total Cost**	**Excess**	**Orders**
29	SILVER	21.63	0.48	0.992	58.66	136,201.98	3.44	13.63
	UICP	30.86	0.78	0.988	58.95	136,335.19	3.69	12.03
	Mean Diff	-9.23	-0.30	0.005	-0.29	-133.21	-0.25	
	p-value	0.00	0.00	0.00	0.00	0.35	0.29	
30	SILVER	108.06	10.43	0.938	85.38	191,125.05	18.94	11.86
	UICP	114.61	11.91	0.930	87.97	194,717.06	21.92	10.03
	Mean Diff	-6.55	-1.48	0.008	-2.59	-3,592.00	-2.98	
	p-value	0.01	0.00	0.00	0.00	0.00	0.00	
31	SILVER	21.27	0.48	0.992	169.21	160,192.39	7.38	15.61
	UICP	33.05	0.83	0.987	171.80	160,432.85	8.00	12.28
	Mean Diff	-11.78	-0.35	0.005	-2.59	-240.47	-0.62	
	p-value	0.00	0.00	0.00	0.00	0.22	0.36	
32	SILVER	112.36	10.13	0.935	243.38	240,106.62	51.85	13.60
	UICP	130.92	13.94	0.917	245.98	250,594.64	58.15	10.41
	Mean Diff	-18.56	-3.81	0.018	-2.61	-10,488.02	-6.30	
	p-value	0.00	0.00	0.00	0.00	0.00	0.00	

TABLE E-2.

CYCLIC DEMAND								
Exp#	**Model**	**ACWTBO**	**ACWT**	**SMA**	**Invest**	**Total Cost**	**Excess**	**Orders**
33	SILVER	157.43	32.35	0.829	17.86	236,569.44	8.80	13.95
	UICP	201.48	51.26	0.768	19.43	256,781.98	15.51	13.43
	Mean Diff	-44.05	-18.91	0.061	-1.57	-20,212.54	-6.70	
	p-value	0.00	0.00	0.00	0.00	0.00	0.00	
34	SILVER	67.78	6.90	0.946	284.94	325,985.05	120.00	19.33
	UICP	155.21	31.06	0.825	319.63	447,160.77	262.24	15.56
	Mean Diff	-87.43	-24.16	0.121	-34.69	-121,175.71	-142.24	
	p-value	0.00	0.00	0.00	0.00	0.00	0.00	
35	SILVER	111.07	15.87	0.909	333.02	400,404.60	185.16	16.62
	UICP	189.45	43.75	0.797	368.38	542,385.74	335.92	14.07
	Mean Diff	-78.39	-27.88	0.111	-35.36	-141,981.14	-150.76	
	p-value	0.00	0.00	0.00	0.00	0.00	0.00	
36	SILVER	158.73	29.39	0.858	404.35	539,966.11	265.99	16.04
	UICP	228.14	60.19	0.762	453.02	720,449.36	441.91	13.78
	Mean Diff	-69.42	-30.80	0.096	-48.67	-180,483.26	-175.92	
	p-value	0.00	0.00	0.00	0.00	0.00	0.00	
37	SILVER	144.85	25.64	0.860	19.12	254,848.28	10.73	15.33
	UICP	212.97	54.83	0.760	20.01	273,429.28	20.63	14.39
	Mean Diff	-68.12	-29.20	0.100	-0.88	-18,581.00	-9.90	
	p-value	0.00	0.00	0.00	0.00	0.00	0.00	
38	SILVER	49.88	3.53	0.967	305.52	342,873.90	161.19	25.43
	UICP	192.89	37.68	0.821	338.19	514,703.40	392.86	17.13
	Mean Diff	-143.01	-34.15	0.146	-32.67	-171,829.50	-231.67	
	p-value	0.00	0.00	0.00	0.00	0.00	0.00	
39	SILVER	89.77	10.44	0.932	355.29	415,727.91	243.82	23.38
	UICP	228.00	51.33	0.795	381.77	616,543.39	453.48	15.33
	Mean Diff	-138.23	-40.89	0.137	-26.48	-200,815.48	-209.66	
	p-value	0.00	0.00	0.00	0.00	0.00	0.00	

TABLE E-2. (CONTINUED)

CYCLIC DEMAND								
Exp#	**Model**	**ACWTBO**	**ACWT**	**SMA**	**Invest**	**Total Cost**	**Excess**	**Orders**
40	SILVER	126.13	18.91	0.902	444.41	543,532.02	340.41	21.80
	UICP	265.37	67.00	0.763	463.74	800,470.45	544.35	15.00
	Mean Diff	-139.24	-48.09	0.138	-19.32	-256,938.42	-203.94	
	p-value	0.00	0.00	0.00	0.00	0.00	0.00	

TABLE E-3.

DECLINING DEMAND								
Exp#	**Model**	**ACWTBO**	**ACWT**	**SMA**	**Invest**	**Total Cost**	**Excess**	**Orders**
41	SILVER	30.69	4.43	0.970	15.13	64,998.35	7.80	3.73
	UICP	20.66	2.82	0.979	18.13	72,268.17	11.52	2.67
	Mean Diff	10.03	1.61	-0.009	-3.00	-7,269.81	-3.72	
	p-value	0.00	0.00	0.00	0.00	0.00	0.00	
42	SILVER	3.90	0.32	0.997	266.88	121,386.40	71.94	5.93
	UICP	8.73	0.66	0.993	344.40	141,286.77	160.87	2.75
	Mean Diff	-4.84	-0.34	0.004	-77.52	-19,900.36	-88.93	
	p-value	0.00	0.04	0.00	0.00	0.00	0.00	
43	SILVER	18.16	2.52	0.981	325.22	144,195.39	124.14	4.89
	UICP	23.58	3.36	0.978	401.73	166,109.74	216.38	2.70
	Mean Diff	-5.42	-0.83	0.003	-76.52	-21,914.35	-92.23	
	p-value	0.03	0.12	0.22	0.00	0.00	0.00	
44	SILVER	34.64	7.29	0.958	430.25	193,247.43	197.15	4.57
	UICP	38.40	7.41	0.958	513.17	216,430.21	300.71	2.62
	Mean Diff	-3.75	-0.12	0.000	-82.92	-23,182.78	-103.57	
	p-value	0.22	0.89	0.89	0.00	0.00	0.00	
45	SILVER	35.95	4.77	0.967	17.17	74,915.90	9.06	4.34
	UICP	27.11	3.17	0.976	19.46	81,361.60	14.18	3.66
	Mean Diff	8.84	1.60	-0.009	-2.29	-6,445.70	-5.12	
	p-value	0.00	0.00	0.00	0.00	0.00	0.00	
46	SILVER	5.57	0.38	0.995	323.75	148,384.31	119.53	8.64
	UICP	13.69	0.96	0.989	400.36	170,606.27	269.31	4.41
	Mean Diff	-8.12	-0.58	0.006	-76.61	-22,221.95	-149.78	
	p-value	0.00	0.00	0.00	0.00	0.00	0.00	
47	SILVER	29.03	3.78	0.975	388.37	177,435.53	176.67	7.36
	UICP	32.65	4.09	0.973	455.37	197,663.46	312.46	4.06
	Mean Diff	-3.62	-0.31	0.002	-67.00	-20,227.93	-135.79	
	p-value	0.22	0.57	0.50	0.00	0.00	0.00	

TABLE E-3. (CONTINUED)

DECLINING DEMAND								
Exp#	**Model**	**ACWTBO**	**ACWT**	**SMA**	**Invest**	**Total Cost**	**Excess**	**Orders**
48	SILVER	44.15	6.91	0.956	488.82	224,530.43	240.94	6.79
	UICP	54.27	8.79	0.950	561.40	251,018.99	385.89	3.83
	Mean Diff	-10.12	-1.87	0.006	-72.58	-26,488.56	-144.95	
	p-value	0.00	0.01	0.03	0.00	0.00	0.00	
49	SILVER	30.43	4.37	0.969	15.83	68,826.68	7.97	4.19
	UICP	21.45	2.66	0.979	18.57	75,696.67	12.12	3.05
	Mean Diff	8.98	1.71	-0.010	-2.74	-6,870.00	-4.16	
	p-value	0.00	0.00	0.00	0.00	0.00	0.00	
50	SILVER	4.61	0.34	0.996	288.14	132,156.26	85.38	7.11
	UICP	12.78	1.02	0.990	359.90	151,128.44	182.93	3.37
	Mean Diff	-8.18	-0.68	0.006	-71.76	-18,972.19	-97.55	
	p-value	0.00	0.00	0.00	0.00	0.00	0.00	
51	SILVER	19.69	2.38	0.981	345.51	154,287.56	133.76	5.74
	UICP	28.58	4.00	0.974	414.78	175,930.68	231.85	3.20
	Mean Diff	-8.89	-1.62	0.007	-69.27	-21,643.11	-98.09	
	p-value	0.00	0.01	0.01	0.00	0.00	0.00	
52	SILVER	42.23	8.10	0.952	450.48	206,360.51	202.72	5.32
	UICP	41.34	8.04	0.956	529.65	229,101.98	320.76	3.10
	Mean Diff	0.89	0.06	-0.003	-79.18	-22,741.47	-118.04	
	p-value	0.78	0.94	0.34	0.00	0.00	0.00	
53	SILVER	38.79	5.14	0.968	17.49	76,847.62	9.20	4.54
	UICP	27.37	3.17	0.975	19.48	82,230.65	13.87	3.81
	Mean Diff	11.43	1.97	-0.007	-2.00	-5,383.03	-4.67	
	p-value	0.00	0.00	0.00	0.00	0.00	0.00	
54	SILVER	6.51	0.43	0.995	333.17	154,374.86	124.29	9.42
	UICP	14.81	1.05	0.989	401.45	172,810.70	270.38	4.59
	Mean Diff	-8.30	-0.63	0.006	-68.28	-18,435.84	-146.08	
	p-value	0.00	0.00	0.00	0.00	0.00	0.00	

TABLE E-3. (CONTINUED)

DECLINING DEMAND								
Exp#	Model	ACWTBO	ACWT	SMA	Invest	Total Cost	Excess	Orders
55	SILVER	25.86	3.75	0.977	397.58	182,631.00	179.41	7.74
	UICP	32.66	3.86	0.973	457.44	198,865.12	308.88	4.23
	Mean Diff	-6.80	-0.11	0.004	-59.86	-16,234.12	-129.47	
	p-value	0.02	0.87	0.07	0.00	0.00	0.00	
56	SILVER	48.65	7.72	0.951	498.29	232,432.58	241.98	7.19
	UICP	54.52	8.85	0.950	562.66	252,866.83	385.09	4.00
	Mean Diff	-5.97	-1.13	0.001	-64.36	-20,434.24	-143.11	
	p-value	0.11	0.17	0.76	0.00	0.00	0.00	
57	SILVER	34.36	4.52	0.969	16.26	71,307.92	8.16	4.38
	UICP	22.56	2.53	0.981	18.76	77,517.60	12.27	3.29
	Mean Diff	11.80	1.99	-0.012	-2.50	-6,209.68	-4.11	
	p-value	0.00	0.00	0.00	0.00	0.00	0.00	
58	SILVER	5.66	0.43	0.995	302.14	139,124.33	98.06	7.71
	UICP	13.86	1.24	0.988	369.65	157,205.19	201.40	3.71
	Mean Diff	-8.21	-0.82	0.007	-67.51	-18,080.86	-103.34	
	p-value	0.00	0.00	0.00	0.00	0.00	0.00	
59	SILVER	22.87	2.88	0.979	361.23	162,142.88	146.52	6.12
	UICP	29.70	3.92	0.972	428.68	183,184.14	252.98	3.50
	Mean Diff	-6.83	-1.05	0.006	-67.45	-21,041.26	-106.46	
	p-value	0.01	0.02	0.01	0.00	0.00	0.00	
60	SILVER	40.49	6.55	0.958	464.34	210,829.51	212.65	5.67
	UICP	40.07	7.42	0.958	535.47	232,694.36	330.43	3.30
	Mean Diff	0.43	-0.87	-0.000	-71.13	-21,864.85	-117.78	
	p-value	0.90	0.28	0.91	0.00	0.00	0.00	
61	SILVER	39.55	5.02	0.969	17.81	78,773.64	9.28	4.72
	UICP	28.36	3.11	0.976	19.55	83,039.20	13.66	3.90
	Mean Diff	11.19	1.92	-0.007	-1.74	-4,265.55	-4.39	
	p-value	0.00	0.00	0.00	0.00	0.00	0.00	

TABLE E-3. (CONTINUED)

DECLINING DEMAND								
Exp#	**Model**	**ACWTBO**	**ACWT**	**SMA**	**Invest**	**Total Cost**	**Excess**	**Orders**
62	SILVER	3.76	0.17	0.997	341.58	158,844.11	131.81	9.84
	UICP	16.90	1.24	0.988	403.99	175,249.35	271.34	4.76
	Mean Diff	-13.15	-1.08	0.010	-62.40	-16,405.24	-139.53	
	p-value	0.00	0.00	0.00	0.00	0.00	0.00	
63	SILVER	25.50	3.38	0.977	406.21	186,671.11	187.18	8.05
	UICP	37.35	5.10	0.968	460.49	204,774.76	313.81	4.37
	Mean Diff	-11.85	-1.72	0.009	-54.28	-18,103.65	-126.63	
	p-value	0.00	0.01	0.00	0.00	0.00	0.00	
64	SILVER	47.50	7.97	0.952	505.52	238,575.80	242.72	7.58
	UICP	55.09	9.65	0.948	567.07	259,222.98	384.43	4.12
	Mean Diff	-7.59	-1.68	0.004	-61.55	-20,647.18	-141.71	
	p-value	0.04	0.07	0.26	0.00	0.00	0.00	
65	SILVER	28.79	4.90	0.967	14.42	60,850.50	10.41	3.23
	UICP	21.59	3.27	0.977	18.47	72,835.42	15.56	2.55
	Mean Diff	7.20	1.63	-0.010	-4.06	-11,984.92	-5.15	
	p-value	0.00	0.00	0.00	0.00	0.00	0.00	
66	SILVER	3.35	0.19	0.997	251.17	114,246.57	137.14	7.08
	UICP	10.11	0.95	0.990	362.27	146,638.34	294.61	2.85
	Mean Diff	-6.76	-0.75	0.007	-111.10	-32,391.78	-157.44	
	p-value	0.00	0.00	0.00	0.00	0.00	0.00	
67	SILVER	21.70	3.37	0.976	316.10	140,592.78	204.26	5.76
	UICP	25.08	3.85	0.976	421.75	171,973.70	355.29	2.74
	Mean Diff	-3.38	-0.47	0.001	-105.64	-31,380.92	-151.04	
	p-value	0.21	0.48	0.77	0.00	0.00	0.00	
68	SILVER	36.07	7.59	0.952	420.45	187,534.24	298.10	6.43
	UICP	37.24	7.98	0.957	531.67	220,175.14	453.74	2.62
	Mean Diff	-1.17	-0.38	-0.006	-111.22	-32,640.90	-155.64	
	p-value	0.73	0.68	0.11	0.00	0.00	0.00	

TABLE E-3. (CONTINUED)

DECLINING DEMAND								
Exp#	**Model**	**ACWTBO**	**ACWT**	**SMA**	**Invest**	**Total Cost**	**Excess**	**Orders**
69	SILVER	35.57	4.23	0.969	16.40	69,742.07	11.01	3.69
	UICP	29.91	3.56	0.975	19.66	81,398.19	17.55	3.52
	Mean Diff	5.66	0.67	-0.006	-3.26	-11,656.12	-6.54	
	p-value	0.06	0.15	0.01	0.00	0.00	0.00	
70	SILVER	5.17	0.47	0.995	300.96	140,351.30	147.22	11.58
	UICP	15.74	1.26	0.987	406.81	173,176.59	376.65	4.44
	Mean Diff	-10.57	-0.79	0.008	-105.85	-32,825.29	-229.42	
	p-value	0.00	0.00	0.00	0.00	0.00	0.00	
71	SILVER	27.93	3.96	0.974	370.34	170,989.74	227.69	10.92
	UICP	39.35	6.33	0.963	461.76	202,788.81	424.49	4.04
	Mean Diff	-11.43	-2.38	0.011	-91.42	-31,799.07	-196.80	
	p-value	0.00	0.00	0.00	0.00	0.00	0.00	
72	SILVER	44.11	7.46	0.955	466.90	217,640.86	305.45	11.19
	UICP	49.40	9.34	0.949	569.10	252,276.62	510.21	3.82
	Mean Diff	-5.29	-1.88	0.006	-102.20	-34,635.76	-204.76	
	p-value	0.15	0.03	0.07	0.00	0.00	0.00	
73	SILVER	27.58	4.41	0.970	15.31	65,293.84	8.32	3.58
	UICP	19.73	2.50	0.982	18.09	71,901.36	11.63	2.61
	Mean Diff	7.85	1.90	-0.011	-2.78	-6,607.52	-3.31	
	p-value	0.00	0.00	0.00	0.00	0.00	0.00	
74	SILVER	3.52	0.32	0.997	275.93	123,063.99	92.87	5.27
	UICP	11.36	1.04	0.990	346.44	142,511.75	168.89	2.74
	Mean Diff	-7.84	-0.72	0.006	-70.51	-19,447.76	-76.02	
	p-value	0.00	0.00	0.00	0.00	0.00	0.00	
75	SILVER	19.69	2.79	0.980	332.45	146,260.47	141.41	4.58
	UICP	26.47	3.51	0.976	404.84	167,303.13	226.11	2.70
	Mean Diff	-6.78	-0.72	0.004	-72.39	-21,042.66	-84.70	
	p-value	0.02	0.22	0.14	0.00	0.00	0.00	

TABLE E-3. (CONTINUED)

DECLINING DEMAND								
Exp#	**Model**	**ACWTBO**	**ACWT**	**SMA**	**Invest**	**Total Cost**	**Excess**	**Orders**
76	SILVER	35.12	6.95	0.962	437.45	193,679.41	220.61	4.37
	UICP	37.58	6.98	0.958	515.62	215,666.03	313.65	2.58
	Mean Diff	-2.46	-0.03	0.003	-78.17	-21,968.62	-93.04	
	p-value	0.45	0.97	0.31	0.00	0.00	0.00	

TABLE E-4.

INCREASING DEMAND								
Exp#	**Model**	**ACWTBO**	**ACWT**	**SMA**	**Invest**	**Total Cost**	**Excess**	**Orders**
77	SILVER	143.07	26.73	0.861	21.78	165,477.18	3.89	11.66
	UICP	184.01	42.83	0.790	20.39	165,311.83	4.08	12.88
	Mean Diff	-40.94	-16.10	0.071	1.40	165.35	-0.19	
	p-value	0.00	0.00	0.00	0.00	0.74	0.44	
78	SILVER	68.30	6.92	0.945	337.30	276,451.50	46.65	19.54
	UICP	156.18	33.74	0.814	316.56	364,407.69	54.30	15.65
	Mean Diff	-87.88	-26.82	0.131	20.74	-87,956.20	-7.66	
	p-value	0.00	0.00	0.00	0.00	0.00	0.03	
79	SILVER	110.49	15.85	0.908	376.97	329,851.16	107.54	15.99
	UICP	186.64	44.23	0.788	354.98	427,423.80	116.00	13.98
	Mean Diff	-76.15	-28.38	0.120	22.00	-97,572.64	-8.46	
	p-value	0.00	0.00	0.00	0.00	0.00	0.06	
80	SILVER	159.83	31.91	0.854	455.54	458,439.78	181.90	14.97
	UICP	237.05	64.61	0.750	429.90	587,491.78	190.86	13.58
	Mean Diff	-77.22	-32.70	0.103	25.64	-129,052.00	-8.97	
	p-value	0.00	0.00	0.00	0.00	0.00	0.10	
81	SILVER	122.25	21.89	0.882	21.27	158,186.27	4.27	11.05
	UICP	212.26	57.33	0.758	19.38	159,642.09	4.06	12.19
	Mean Diff	-90.01	-35.43	0.124	1.88	-1,455.82	0.21	
	p-value	0.00	0.00	0.00	0.00	0.02	0.43	
82	SILVER	49.59	4.66	0.959	320.02	254,033.34	39.84	19.18
	UICP	197.77	49.01	0.778	299.11	391,571.54	54.76	15.11
	Mean Diff	-148.18	-44.35	0.182	20.92	-137,538.20	-14.92	
	p-value	0.00	0.00	0.00	0.00	0.00	0.00	
83	SILVER	87.23	12.40	0.922	357.24	301,594.98	95.41	16.09
	UICP	235.07	66.87	0.739	335.08	479,368.38	105.39	13.53
	Mean Diff	-147.83	-54.47	0.183	22.16	-177,773.40	-9.98	
	p-value	0.00	0.00	0.00	0.00	0.00	0.03	

TABLE E-4. (CONTINUED)

INCREASING DEMAND								
Exp#	**Model**	**ACWTBO**	**ACWT**	**SMA**	**Invest**	**Total Cost**	**Excess**	**Orders**
84	SILVER	124.27	21.85	0.890	434.72	395,656.47	163.67	15.42
	UICP	268.17	83.44	0.709	406.00	627,491.94	176.57	13.31
	Mean Diff	-143.90	-61.59	0.181	28.72	-231,835.47	-12.91	
	p-value	0.00	0.00	0.00	0.00	0.00	0.03	
85	SILVER	122.05	17.59	0.897	25.04	183,019.74	4.24	12.53
	UICP	191.78	37.61	0.822	23.01	181,494.89	4.23	13.08
	Mean Diff	-69.74	-20.02	0.075	2.03	1,524.85	0.01	
	p-value	0.00	0.00	0.00	0.00	0.01	0.96	
86	SILVER	40.41	2.50	0.974	388.54	291,971.67	44.68	21.46
	UICP	167.59	28.59	0.846	362.62	386,220.83	53.61	15.50
	Mean Diff	-127.17	-26.09	0.128	25.92	-94,249.16	-8.93	
	p-value	0.00	0.00	0.00	0.00	0.00	0.01	
87	SILVER	77.90	8.29	0.944	431.66	338,744.37	97.85	18.02
	UICP	199.19	39.05	0.821	401.32	457,520.17	104.92	14.02
	Mean Diff	-121.29	-30.76	0.124	30.34	-118,775.81	-7.08	
	p-value	0.00	0.00	0.00	0.00	0.00	0.14	
88	SILVER	125.04	18.39	0.904	522.99	443,260.67	165.96	16.66
	UICP	232.23	54.56	0.785	488.50	603,029.44	172.19	13.55
	Mean Diff	-107.19	-36.17	0.119	34.49	-159,768.77	-6.23	
	p-value	0.00	0.00	0.00	0.00	0.00	0.33	
89	SILVER	103.85	15.39	0.912	22.75	165,320.91	4.02	11.50
	UICP	202.78	52.40	0.770	20.24	164,614.96	4.19	12.38
	Mean Diff	-98.93	-37.01	0.141	2.50	705.95	-0.17	
	p-value	0.00	0.00	0.00	0.00	0.31	0.49	
90	SILVER	31.78	2.28	0.976	344.19	260,046.80	43.04	19.68
	UICP	207.01	48.14	0.788	315.67	408,421.43	53.91	14.84
	Mean Diff	-175.23	-45.86	0.188	28.52	-148,374.63	-10.87	
	p-value	0.00	0.00	0.00	0.00	0.00	0.00	

TABLE E-4. (CONTINUED)

INCREASING DEMAND								
Exp#	**Model**	**ACWTBO**	**ACWT**	**SMA**	**Invest**	**Total Cost**	**Excess**	**Orders**
91	SILVER	67.28	7.60	0.947	383.09	300,489.38	101.21	16.85
	UICP	236.37	60.66	0.766	349.83	479,226.44	109.33	13.62
	Mean Diff	-169.09	-52.76	0.180	33.25	-178,737.05	-8.12	
	p-value	0.00	0.00	0.00	0.00	0.00	0.08	
92	SILVER	106.28	17.03	0.907	457.10	388,288.16	158.15	16.07
	UICP	269.62	80.47	0.726	416.40	633,183.18	169.64	13.64
	Mean Diff	-163.34	-63.44	0.181	40.70	-244,895.02	-11.49	
	p-value	0.00	0.00	0.00	0.00	0.00	0.05	

TABLE E-5.

CYCLIC DEMAND - FIXED FORECASTS								
Exp#	**Model**	**ACWTBO**	**ACWT**	**SMA**	**Invest**	**Total Cost**	**Excess**	**Orders**
33F	SILVER	194.96	52.02	0.757	19.27	256,742.51	15.37	19.27
	UICP	201.48	51.26	0.768	19.43	256,781.98	15.51	13.43
	Mean Diff	-6.52	0.76	-0.011	-0.16	-39.47	-0.14	
	p-value	0.06	0.57	0.01	0.01	0.95	0.44	
34F	SILVER	137.03	24.98	0.842	316.61	426,611.15	254.81	21.81
	UICP	155.21	31.06	0.825	319.63	447,160.77	262.24	15.56
	Mean Diff	-18.18	-6.07	0.017	-3.02	-20,549.61	-7.43	
	p-value	0.00	0.00	0.00	0.00	0.00	0.00	
35F	SILVER	168.58	38.06	0.808	362.06	518,729.15	320.44	19.31
	UICP	189.45	43.75	0.797	368.38	542,385.74	335.92	14.07
	Mean Diff	-20.88	-5.68	0.010	-6.33	-23,656.58	-15.48	
	p-value	0.00	0.00	0.00	0.00	0.00	0.00	
36F	SILVER	218.81	59.65	0.764	434.78	696,485.63	411.86	18.76
	UICP	228.14	60.19	0.762	453.02	720,449.36	441.91	13.78
	Mean Diff	-9.34	-3.24	0.002	-18.24	-23,963.74	-30.05	
	p-value	0.00	0.02	0.51	0.00	0.00	0.00	
37F	SILVER	211.45	55.59	0.760	20.02	275,452.97	20.46	16.06
	UICP	212.97	54.83	0.760	20.01	273,429.28	20.63	14.39
	Mean Diff	-1.51	0.75	0.000	0.018	2,023.69	-0.17	
	p-value	0.72	0.63	0.97	0.76	0.00	0.39	
38F	SILVER	176.29	31.84	0.837	336.50	494,377.71	388.48	23.90
	UICP	192.89	37.68	0.821	338.19	514,703.40	392.86	17.13
	Mean Diff	-16.60	-5.84	0.016	-1.69	-20,325.69	-4.38	
	p-value	0.00	0.00	0.00	0.00	0.00	0.06	
39F	SILVER	210.78	45.68	0.806	377.04	592,899.11	440.74	21.12
	UICP	228.00	51.33	0.795	381.77	616,543.39	453.48	15.33
	Mean Diff	-17.22	-5.65	0.011	-4.73	-23,644.28	-12.73	
	p-value	0.00	0.00	0.00	0.00	0.00	0.00	

TABLE E-5. (CONTINUED)

CYCLIC DEMAND - FIXED FORECASTS								
Exp#	**Model**	**ACWTBO**	**ACWT**	**SMA**	**Invest**	**Total Cost**	**Excess**	**Orders**
40F	SILVER	249.20	59.81	0.774	459.53	767,418.35	532.71	20.50
	UICP	265.37	67.00	0.763	463.74	800,470.45	544.35	15.00
	Mean Diff	-16.17	-7.19	0.011	-4.20	-33,052.10	-11.64	
	p-value	0.00	0.00	0.00	0.00	0.00	0.00	

TABLE E-6.

DECLINING DEMAND - FIXED FORECASTS								
Exp#	**Model**	**ACWTBO**	**ACWT**	**SMA**	**Invest**	**Total Cost**	**Excess**	**Orders**
41F	SILVER	28.74	4.20	0.972	18.33	73,010.58	11.78	2.60
	UICP	20.66	2.82	0.979	18.13	72,268.17	11.52	2.67
	Mean Diff	8.08	1.38	-0.007	0.20	742.42	0.27	
	p-value	0.00	0.00	0.00	0.07	0.04	0.09	
42F	SILVER	3.90	0.34	0.997	332.88	138,748.64	147.20	4.99
	UICP	8.73	0.66	0.993	344.40	141,286.77	160.87	2.75
	Mean Diff	-4.84	-0.32	0.004	-11.53	-2,538.12	-13.66	
	p-value	0.00	0.06	0.00	0.00	0.00	0.00	
43F	SILVER	15.35	2.06	0.984	392.79	162,140.48	208.20	4.36
	UICP	23.58	3.36	0.978	401.73	166,109.74	216.38	2.70
	Mean Diff	-8.24	-1.30	0.006	-8.94	-3,969.26	-8.18	
	p-value	0.00	0.01	0.01	0.00	0.00	0.00	
44F	SILVER	30.50	6.53	0.962	499.17	210,560.98	286.78	4.04
	UICP	38.40	7.41	0.958	513.17	216,430.21	300.71	2.62
	Mean Diff	-7.90	-0.88	0.004	-14.00	-5,869.24	-13.93	
	p-value	0.01	0.27	0.26	0.00	0.00	0.00	
45F	SILVER	33.44	4.20	0.970	19.63	81,985.05	14.32	3.62
	UICP	27.11	3.17	0.976	19.46	81,361.60	14.18	3.66
	Mean Diff	6.34	1.03	-0.006	0.17	623.45	0.14	
	p-value	0.02	0.01	0.00	0.06	0.04	0.37	
46F	SILVER	5.23	0.36	0.995	388.47	168,618.61	256.26	8.10
	UICP	13.69	0.96	0.989	400.36	170,606.27	269.31	4.41
	Mean Diff	-8.46	-0.60	0.006	-11.89	-1,987.66	-13.05	
	p-value	0.00	0.00	0.00	0.00	0.00	0.00	
47F	SILVER	26.29	3.39	0.976	445.69	194,445.58	302.38	6.59
	UICP	32.65	4.09	0.973	455.37	197,663.46	312.46	4.06
	Mean Diff	-6.36	-0.70	0.003	-9.68	-3,217.89	-10.08	
	p-value	0.03	0.20	0.20	0.00	0.01	0.00	

TABLE E-6. (CONTINUED)

DECLINING DEMAND - FIXED FORECASTS								
Exp#	**Model**	**ACWTBO**	**ACWT**	**SMA**	**Invest**	**Total Cost**	**Excess**	**Orders**
48F	SILVER	39.92	6.37	0.959	548.44	241,665.08	371.60	6.02
	UICP	54.27	8.79	0.950	561.40	251,018.99	385.89	3.83
	Mean Diff	-14.35	-2.42	0.008	-12.96	-9,353.91	-14.30	
	p-value	0.00	0.00	0.00	0.00	0.00	0.00	
49F	SILVER	28.54	3.64	0.974	18.72	76,028.43	12.23	2.97
	UICP	21.45	2.66	0.979	18.57	75,696.67	12.12	3.05
	Mean Diff	7.08	0.98	-0.005	0.15	331.76	0.10	
	p-value	0.01	0.01	0.01	0.14	0.33	0.51	
50F	SILVER	3.48	0.25	0.997	348.19	148,052.60	171.21	6.11
	UICP	12.78	1.02	0.990	359.90	151,128.44	182.93	3.37
	Mean Diff	-9.30	-0.77	0.007	-11.71	-3,075.84	-11.72	
	p-value	0.00	0.00	0.00	0.00	0.00	0.00	
51F	SILVER	16.75	1.98	0.983	406.45	170,695.64	224.47	5.20
	UICP	28.58	4.00	0.974	414.78	175,930.68	231.85	3.20
	Mean Diff	-11.83	-2.01	0.009	-8.33	-5,235.03	-7.38	
	p-value	0.00	0.00	0.00	0.00	0.00	0.00	
52F	SILVER	33.81	6.19	0.961	514.65	220,562.20	305.46	4.8
	UICP	41.34	8.04	0.956	529.65	229,101.98	320.76	3.10
	Mean Diff	-7.53	-1.85	0.005	-15.00	-8,539.78	-15.30	
	p-value	0.02	0.02	0.09	0.00	0.00	0.00	
53F	SILVER	35.41	4.67	0.970	19.66	82,808.43	14.14	3.78
	UICP	27.37	3.17	0.975	19.48	82,230.65	13.87	3.81
	Mean Diff	8.05	1.49	-0.005	0.18	577.79	0.28	
	p-value	0.01	0.00	0.01	0.05	0.05	0.10	
54F	SILVER	6.17	0.41	0.995	390.01	170,678.67	257.47	8.44
	UICP	14.81	1.05	0.989	401.45	172,810.70	270.38	4.54
	Mean Diff	-8.63	-0.65	0.007	-11.44	-2,132.03	-12.90	
	p-value	0.00	0.00	0.00	0.00	0.00	0.00	

TABLE E-6. (CONTINUED)

DECLINING DEMAND - FIXED FORECASTS								
Exp#	**Model**	**ACWTBO**	**ACWT**	**SMA**	**Invest**	**Total Cost**	**Excess**	**Orders**
55F	SILVER	22.74	3.28	0.979	447.24	196,116.06	298.67	6.88
	UICP	32.66	3.86	0.973	457.44	198,865.12	308.88	4.23
	Mean Diff	-9.92	-0.58	0.006	-10.20	-2,749.06	-10.21	
	p-value	0.00	0.39	0.01	0.00	0.06	0.00	
56F	SILVER	43.59	6.96	0.953	549.86	245,542.79	376.02	6.28
	UICP	54.62	8.85	0.950	562.66	252,866.83	385.09	4.00
	Mean Diff	-11.03	-1.89	0.004	-12.80	-7,324.04	-9.07	
	p-value	0.00	0.02	0.27	0.00	0.00	0.01	
57F	SILVER	29.54	3.52	0.975	18.94	78,083.31	12.52	3.21
	UICP	22.56	2.53	0.981	18.76	77,517.60	12.27	3.29
	Mean Diff	6.98	0.99	-0.006	0.18	565.71	0.25	
	p-value	0.01	0.01	0.00	0.06	0.07	0.16	
58F	SILVER	4.43	0.32	0.996	360.07	154,634.03	194.19	6.81
	UICP	13.86	1.24	0.988	369.65	157,205.19	201.40	3.71
	Mean Diff	-9.43	-0.92	0.008	-9.58	-2,571.16	-7.21	
	p-value	0.00	0.00	0.00	0.00	0.00	0.00	
59F	SILVER	19.96	2.44	0.981	419.36	178,139.78	245.06	5.70
	UICP	29.70	3.92	0.972	428.68	183,184.14	252.98	3.50
	Mean Diff	-9.74	-1.48	0.009	-9.32	-5,044.36	-7.92	
	p-value	0.00	0.00	0.00	0.00	0.00	0.00	
60F	SILVER	36.42	6.20	0.961	524.41	227,242.91	318.93	5.23
	UICP	40.07	7.42	0.958	535.47	232,694.36	330.43	3.30
	Mean Diff	-3.65	-1.23	0.003	-11.05	-5,451.45	-11.50	
	p-value	0.27	0.16	0.40	0.00	0.01	0.00	
61F	SILVER	36.86	4.55	0.971	19.74	83,632.29	13.95	3.88
	UICP	28.36	3.11	0.976	19.55	83,039.20	13.66	3.90
	Mean Diff	8.50	1.44	-0.005	0.19	593.10	0.28	
	p-value	0.01	0.00	0.02	0.03	0.04	0.11	

TABLE E-6. (CONTINUED)

DECLINING DEMAND - FIXED FORECASTS								
Exp#	Model	ACWTBO	ACWT	SMA	Invest	Total Cost	Excess	Orders
62F	SILVER	3.45	0.15	0.998	392.86	172,870.06	259.40	8.78
	UICP	16.90	1.24	0.988	403.99	175,249.35	271.34	4.76
	Mean Diff	-13.45	-1.10	0.010	-11.13	-2,379.29	-11.93	
	p-value	0.00	0.00	0.00	0.00	0.00	0.00	
63F	SILVER	24.04	3.06	0.979	450.25	198,362.38	303.15	7.13
	UICP	37.35	5.10	0.968	460.49	204,774.76	313.81	4.37
	Mean Diff	-13.30	-2.04	0.011	-10.24	-6,412.38	-10.66	
	p-value	0.00	0.00	0.00	0.00	0.00	0.00	
64F	SILVER	43.71	7.25	0.955	554.71	250,791.51	374.42	6.52
	UICP	55.09	9.65	0.948	567.07	259,222.98	384.43	4.12
	Mean Diff	-11.38	-2.40	0.007	-12.35	-8,431.48	-10.01	
	p-value	0.00	0.01	0.04	0.00	0.00	0.01	
65F	SILVER	27.53	4.79	0.971	18.63	73,269.15	15.69	2.48
	UICP	21.59	3.27	0.977	18.47	72,835.42	15.56	2.55
	Mean Diff	5.94	1.52	-0.007	0.15	433.73	0.13	
	p-value	0.02	0.01	0.00	0.16	0.23	0.40	
66F	SILVER	3.05	0.16	0.997	349.99	142,963.17	279.85	5.18
	UICP	10.11	0.95	0.990	362.27	146,638.34	294.61	2.85
	Mean Diff	-7.06	-0.78	0.007	-12.28	-3,675.17	-14.77	
	p-value	0.00	0.00	0.00	0.00	0.00	0.00	
67F	SILVER	20.36	3.16	0.979	412.33	169,045.16	345.46	4.44
	UICP	25.08	3.85	0.976	421.75	171,973.70	355.29	2.74
	Mean Diff	-4.72	-0.68	0.004	-9.41	-2,928.54	-9.84	
	p-value	0.07	0.29	0.21	0.00	0.02	0.00	
68F	SILVER	31.99	6.80	0.956	517.12	214,026.58	438.60	4.05
	UICP	37.24	7.98	0.957	531.67	220,175.14	453.74	2.62
	Mean Diff	-5.25	-1.18	-0.001	-14.55	-6,148.56	-15.14	
	p-value	0.10	0.16	0.78	0.00	0.00	0.00	

TABLE E-6. (CONTINUED)

DECLINING DEMAND - FIXED FORECASTS								
Exp#	**Model**	**ACWTBO**	**ACWT**	**SMA**	**Invest**	**Total Cost**	**Excess**	**Orders**
69F	SILVER	35.36	4.05	0.970	19.82	81,946.76	17.65	3.58
	UICP	29.91	3.56	0.975	19.66	81,398.19	17.55	3.52
	Mean Diff	5.45	0.50	-0.005	0.17	548.57	0.10	
	p-value	0.05	0.25	0.02	0.07	0.08	0.55	
70F	SILVER	4.69	0.41	0.996	395.60	171,026.81	364.75	8.18
	UICP	15.74	1.26	0.987	406.81	173,176.59	376.65	4.44
	Mean Diff	-11.05	-0.86	0.009	-11.20	-2,149.78	-11.90	
	p-value	0.00	0.00	0.00	0.00	0.00	0.00	
71F	SILVER	24.03	3.49	0.976	452.44	195,547.12	412.87	6.59
	UICP	39.35	6.33	0.963	461.76	202,788.71	424.49	4.40
	Mean Diff	-15.33	-2.84	0.013	-9.32	-7,241.70	-11.62	
	p-value	0.00	0.00	0.00	0.00	0.00	0.00	
72F	SILVER	37.93	6.72	0.958	556.25	244,217.60	498.46	6.01
	UICP	49.40	9.34	0.949	569.10	252,276.62	510.21	3.82
	Mean Diff	-11.48	-2.62	0.009	-12.85	-8,059.03	-11.75	
	p-value	0.00	0.00	0.01	0.00	0.00	0.00	
73F	SILVER	28.77	4.22	0.972	18.26	72,613.38	11.82	2.54
	UICP	19.73	2.50	0.982	18.09	71,901.36	11.63	2.61
	Mean Diff	9.04	1.72	-0.010	0.17	712.02	0.19	
	p-value	0.00	0.00	0.00	0.11	0.04	0.21	
74F	SILVER	3.62	0.36	0.997	335.22	139,509.43	157.38	5.00
	UICP	11.36	1.04	0.990	346.44	142,511.75	168.89	2.74
	Mean Diff	-7.74	-0.68	0.006	-11.22	-3,002.32	-11.51	
	p-value	0.00	0.00	0.00	0.00	0.00	0.00	
75F	SILVER	17.31	2.42	0.983	394.28	163,143.10	214.33	4.33
	UICP	26.47	3.51	0.976	404.84	167,303.13	226.11	2.70
	Mean Diff	-9.16	-1.08	0.007	-10.56	-4,160.03	-11.78	
	p-value	0.00	0.05	0.02	0.00	0.00	0.00	

TABLE E-6. (CONTINUED)

DECLINING DEMAND - FIXED FORECASTS								
Exp#	**Model**	**ACWTBO**	**ACWT**	**SMA**	**Invest**	**Total Cost**	**Excess**	**Orders**
76F	SILVER	32.63	6.05	0.965	501.08	209,732.14	297.24	3.99
	UICP	37.58	6.98	0.958	515.62	215,666.03	313.65	2.58
	Mean Diff	-4.95	-0.93	0.007	-14.53	-5,933.89	-16.41	
	p-value	0.12	0.24	0.03	0.00	0.00	0.00	

TABLE E-7.

INCREASING DEMAND - FIXED FORECASTS								
Exp#	**Model**	**ACWTBO**	**ACWT**	**SMA**	**Invest**	**Total Cost**	**Excess**	**Orders**
77F	SILVER	188.73	49.01	0.770	20.23	166,059.48	3.67	13.31
	UICP	184.01	42.83	0.790	20.39	165,311.83	4.08	12.88
	Mean Diff	4.72	6.18	-0.020	-0.15	747.64	-0.41	
	p-value	0.21	0.00	0.00	0.01	0.09	0.07	
78F	SILVER	136.14	25.25	0.843	313.25	337,086.99	48.09	21.93
	UICP	156.18	33.74	0.814	316.56	364,407.69	54.30	15.65
	Mean Diff	-20.04	-8.49	0.029	-3.31	-27,320.71	-6.21	
	p-value	0.00	0.00	0.00	0.00	0.00	0.09	
79F	SILVER	168.56	36.63	0.809	350.85	400,769.19	101.16	18.43
	UICP	186.64	44.23	0.788	354.98	427,423.80	116.00	13.98
	Mean Diff	-18.08	-7.60	0.021	-4.13	-26,654.61	-14.84	
	p-value	0.00	0.00	0.00	0.00	0.00	0.00	
80F	SILVER	222.10	57.62	0.763	425.42	561,624.64	183.58	17.54
	UICP	237.05	64.61	0.750	429.90	587,491.78	190.86	13.58
	Mean Diff	-14.95	-6.99	0.013	-4.48	-25,867.13	-7.29	
	p-value	0.00	0.00	0.00	0.00	0.00	0.21	
81F	SILVER	210.91	58.58	0.752	19.34	159,909.78	4.04	12.42
	UICP	212.26	57.33	0.758	19.38	159,642.09	4.06	12.19
	Mean Diff	-1.35	1.26	-0.006	-0.04	267.69	-0.02	
	p-value	0.75	0.46	0.17	0.51	0.58	0.93	
82F	SILVER	182.53	42.05	0.797	296.71	371,948.80	40.74	20.50
	UICP	197.77	49.01	0.778	299.11	391,571.54	54.76	15.11
	Mean Diff	-15.25	-6.96	0.020	-2.40	-19,622.74	-14.02	
	p-value	0.00	0.00	0.00	0.00	0.00	0.00	
83F	SILVER	219.82	59.87	0.754	331.75	457,844.49	94.80	17.28
	UICP	235.07	66.87	0.739	335.08	479,368.38	105.39	13.53
	Mean Diff	-15.25	-7.01	0.014	-3.33	-21,523.89	-10.59	
	p-value	0.00	0.00	0.00	0.00	0.00	0.03	

TABLE E-7. (CONTINUED)

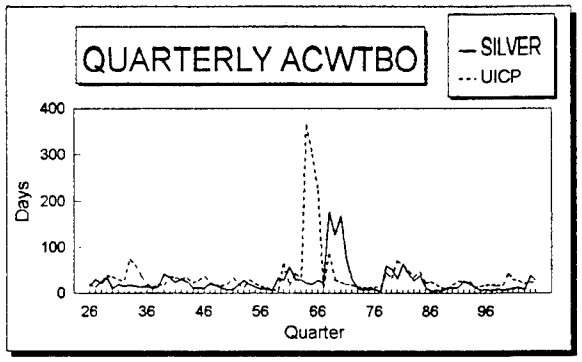
INCREASING DEMAND - FIXED FORECASTS								
Exp#	**Model**	**ACWTBO**	**ACWT**	**SMA**	**Invest**	**Total Cost**	**Excess**	**Orders**
84F	SILVER	249.23	73.07	0.723	400.87	590,587.44	171.12	16.86
	UICP	268.17	83.44	0.709	406.00	627,491.94	176.57	13.31
	Mean Diff	-18.94	-10.36	0.014	-5.13	-36,907.50	-5.46	
	p-value	0.00	0.00	0.00	0.00	0.00	0.39	
85F	SILVER	184.64	39.20	0.810	22.72	181,308.28	4.00	14.13
	UICP	191.78	37.61	0.822	23.01	181,494.89	4.23	13.08
	Mean Diff	-7.14	1.60	-0.013	-0.28	-186.61	-0.22	
	p-value	0.07	0.14	0.00	0.00	0.67	0.37	
86F	SILVER	148.95	21.71	0.873	357.35	362,064.23	45.56	23.25
	UICP	167.59	28.59	0.846	362.62	386,220.83	53.61	15.50
	Mean Diff	-18.64	-6.88	0.027	-5.27	-24,156.60	-8.05	
	p-value	0.00	0.00	0.00	0.00	0.00	0.02	
87F	SILVER	183.24	33.15	0.837	396.34	434,965.07	96.33	19.62
	UICP	199.19	39.05	0.821	401.32	457,520.17	104.92	14.02
	Mean Diff	-15.95	-5.90	0.016	-4.97	-22,555.10	-8.60	
	p-value	0.00	0.00	0.00	0.00	0.00	0.08	
88F	SILVER	217.75	48.06	0.798	481.27	572,939.42	164.39	18.28
	UICP	232.23	54.56	0.785	488.50	603,029.44	172.19	13.55
	Mean Diff	-14.49	-6.50	0.014	-7.22	-30,090.01	-7.80	
	p-value	0.00	0.00	0.00	0.00	0.00	0.22	
89F	SILVER	201.14	54.69	0.763	20.10	165,295.50	4.19	12.90
	UICP	202.78	52.40	0.770	20.24	164,614.96	4.19	12.38
	Mean Diff	-1.64	2.29	-0.007	-0.15	680.54	0.00	
	p-value	0.71	0.16	0.10	0.01	0.15	0.99	
90F	SILVER	192.57	41.38	0.806	312.23	388,358.19	40.14	20.80
	UICP	207.01	48.14	0.788	315.67	408,421.43	53.91	14.84
	Mean Diff	-14.43	-6.76	0.018	-3.44	-20,063.25	-13.77	
	p-value	0.00	0.00	0.00	0.00	0.00	0.00	

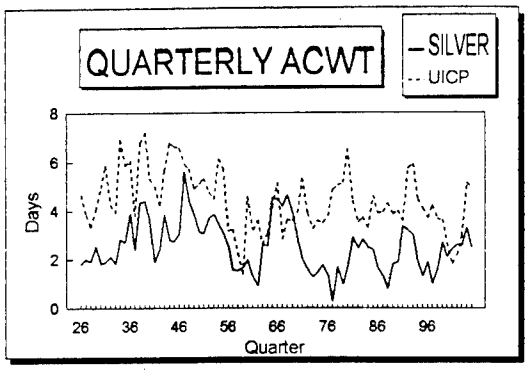
TABLE E-7. (CONTINUED)

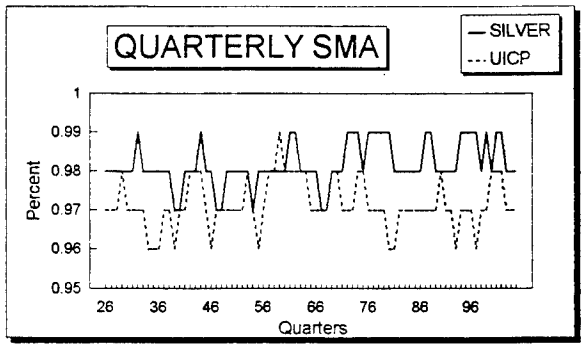
INCREASING DEMAND - FIXED FORECASTS								
Exp#	**Model**	**ACWTBO**	**ACWT**	**SMA**	**Invest**	**Total Cost**	**Excess**	**Orders**
91F	SILVER	222.86	53.48	0.786	346.53	456,851.18	102.02	17.87
	UICP	236.37	60.66	0.766	349.83	479,226.44	109.33	13.62
	Mean Diff	-13.52	-7.18	0.019	-3.30	-22,375.26	-7.31	
	p-value	0.00	0.00	0.00	0.00	0.00	0.13	
92F	SILVER	253.77	74.98	0.732	411.77	614,744.98	150.83	17.41
	UICP	269.62	80.47	0.726	416.40	633,183.18	169.64	13.64
	Mean Diff	-15.85	-5.49	0.006	-4.63	-18,438.20	-18.81	
	p-value	0.00	0.01	0.19	0.00	0.02	0.00	

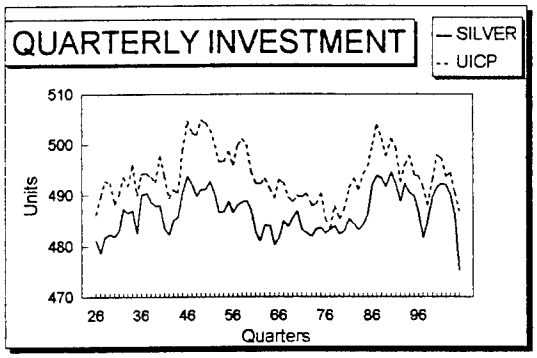
APPENDIX F. STATIONARY DEMAND GRAPHS

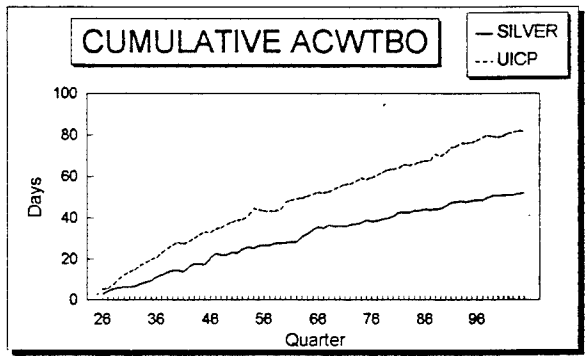
Description: Steady State Experiment # 10

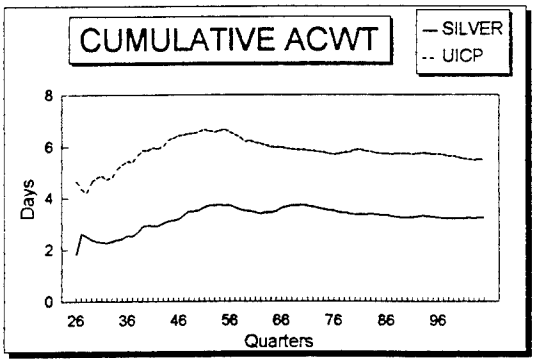


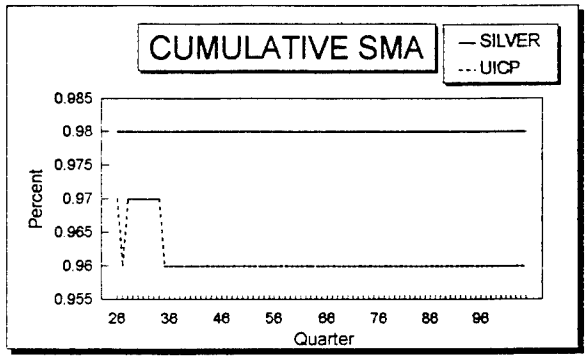


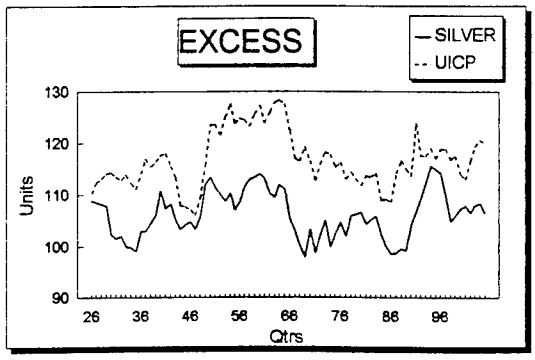






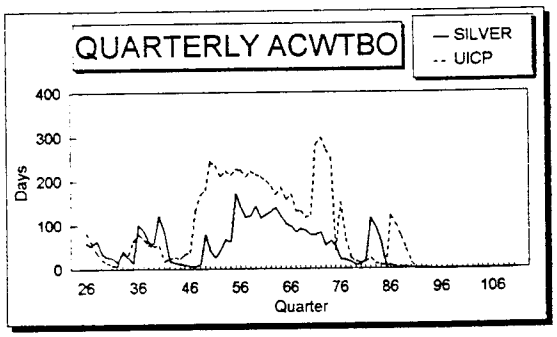


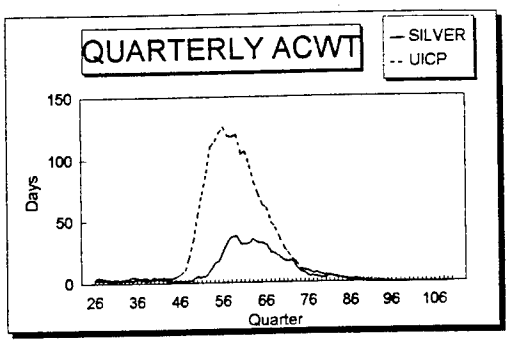


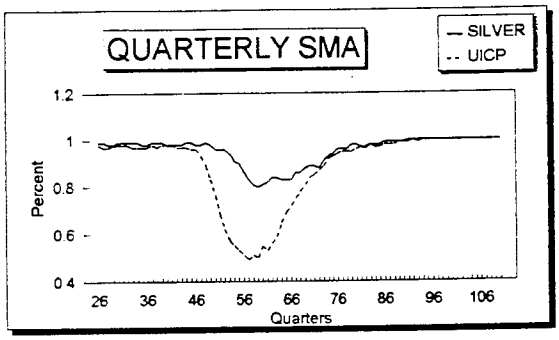


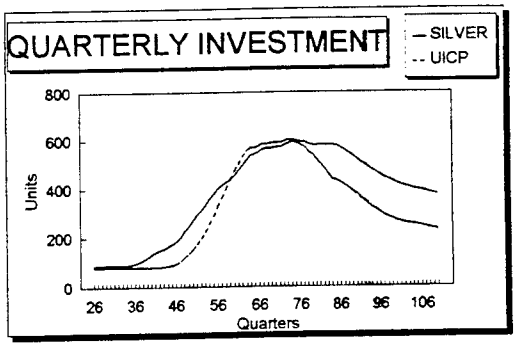
APPENDIX G. CYCLIC DEMAND GRAPHS

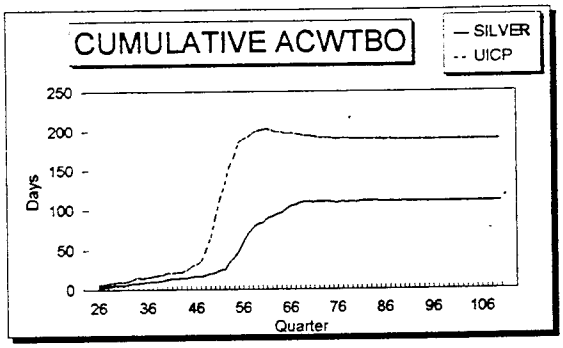
Description: Cyclic Experiment # 35

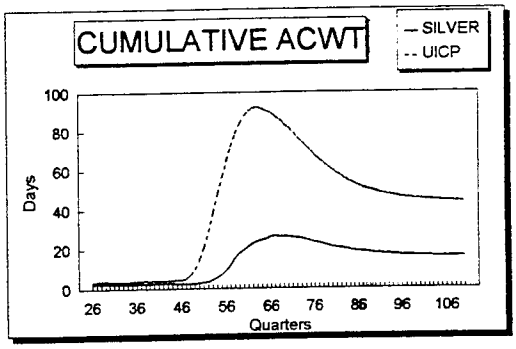


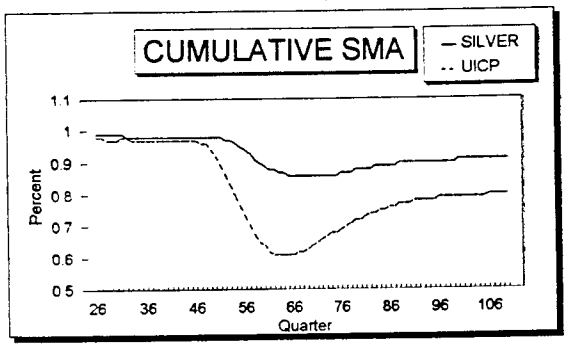


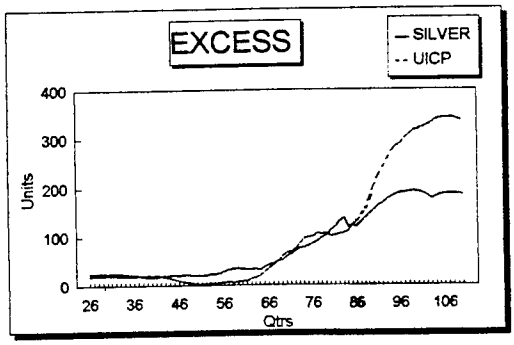






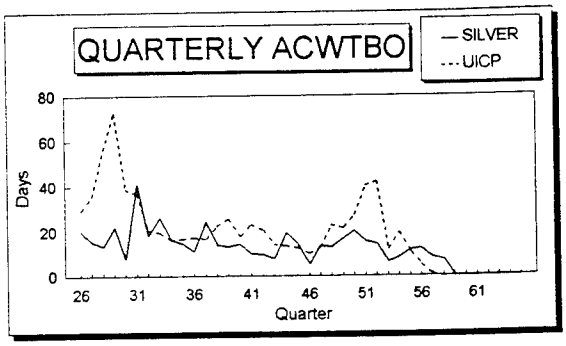


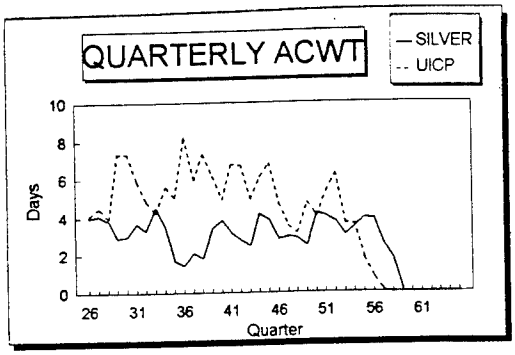


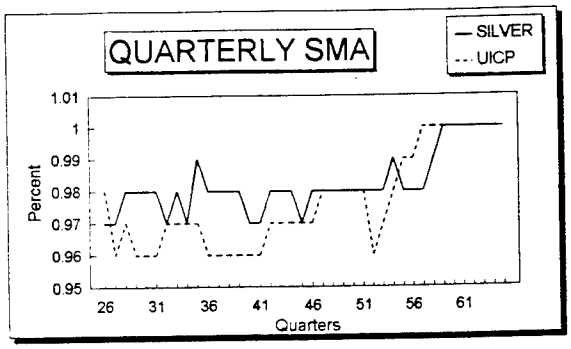


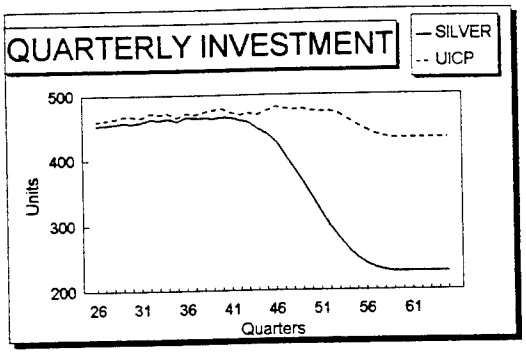
APPENDIX H. DECLINING DEMAND (STEEP TREND) GRAPHS

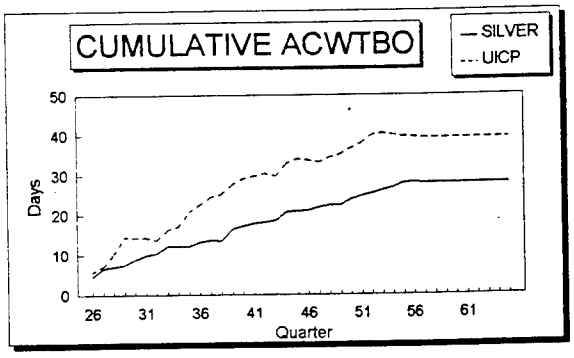
Description: Declining Experiment # 71

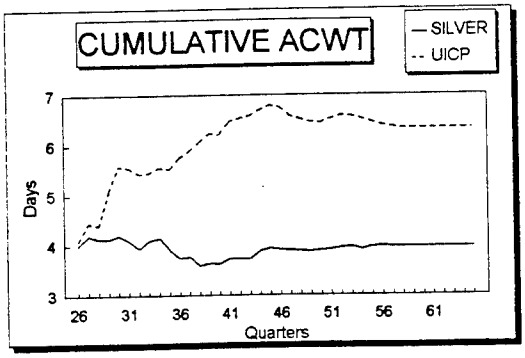


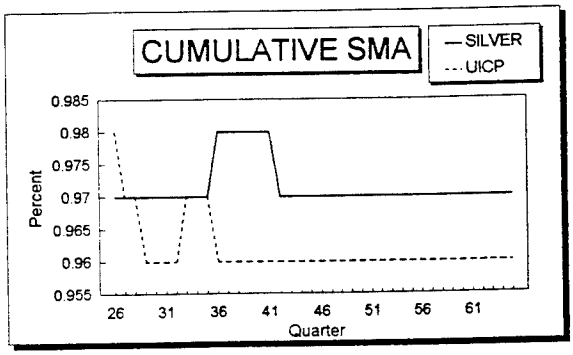


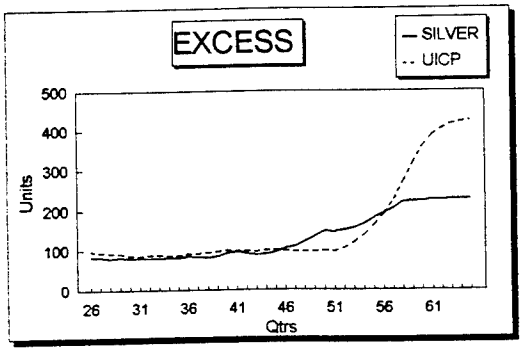






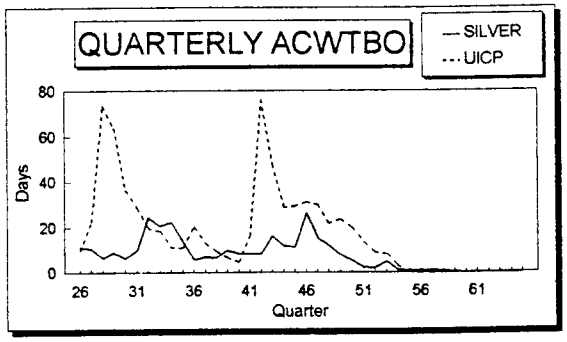


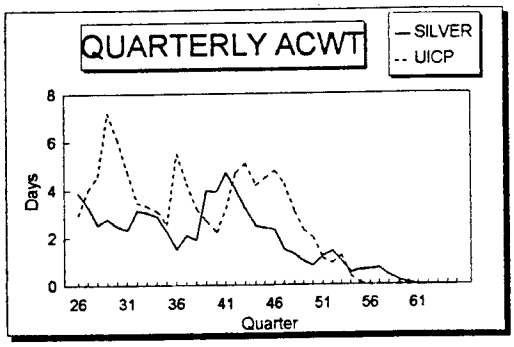


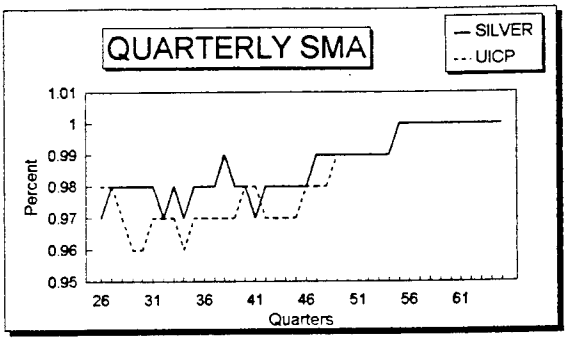


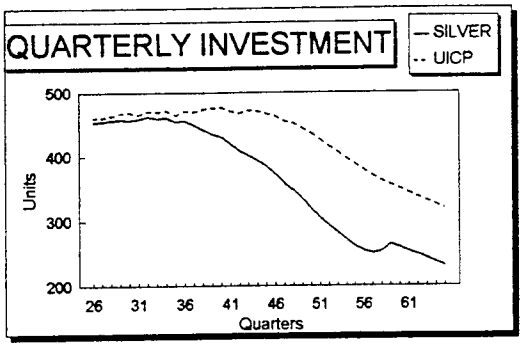
APPENDIX I. DECLINING DEMAND (SLOW TREND) GRAPHS

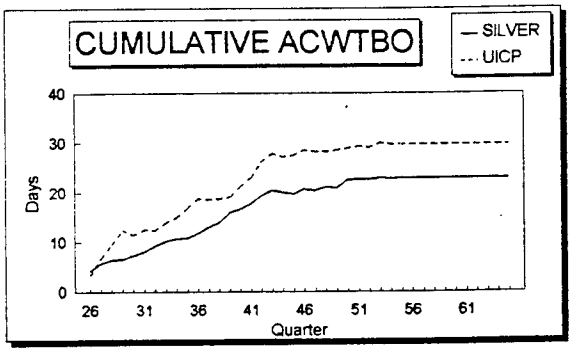
Description: Declining Experiment # 59

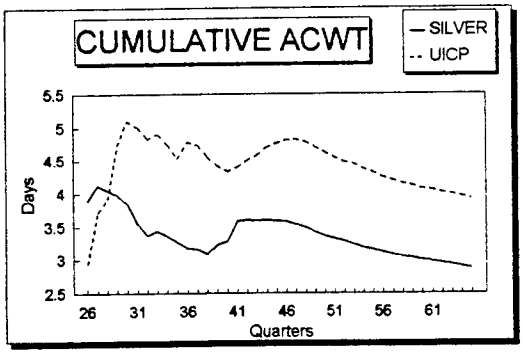


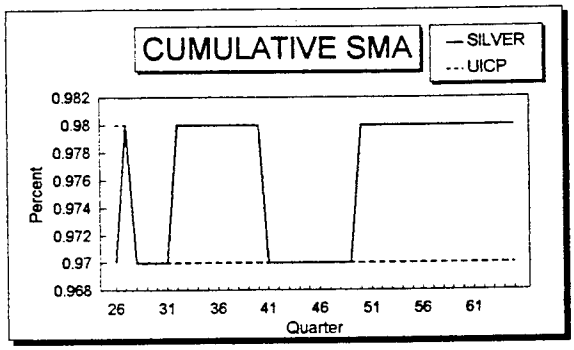


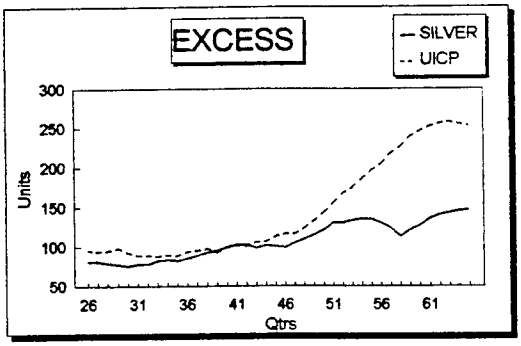






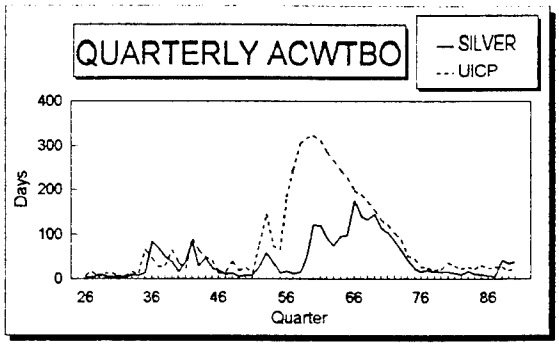


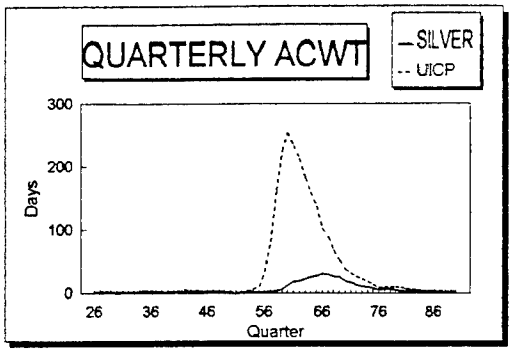


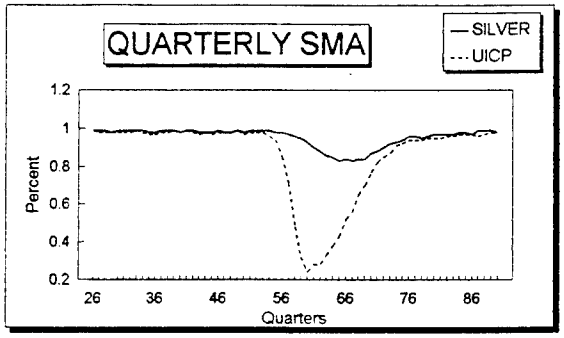


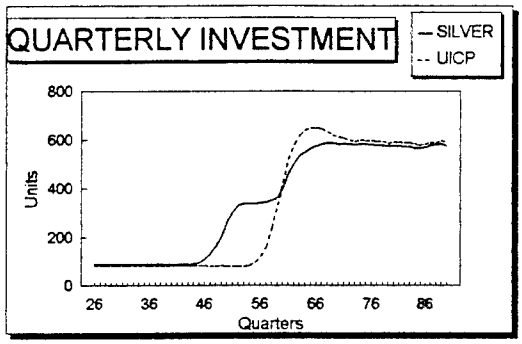
APPENDIX J. INCREASING DEMAND (STEEP TREND) GRAPHS

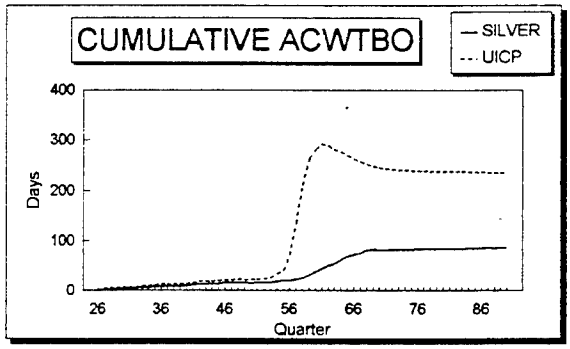
Description: Increasing Experiment # 83

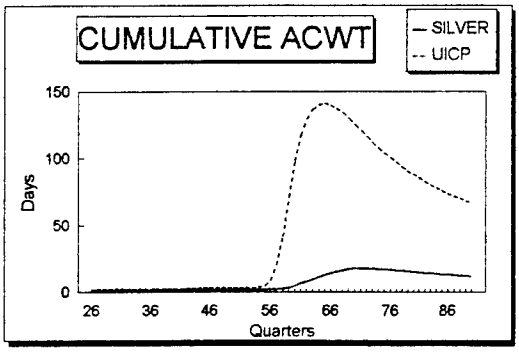


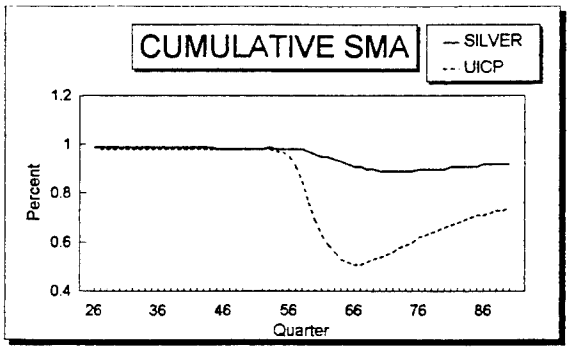


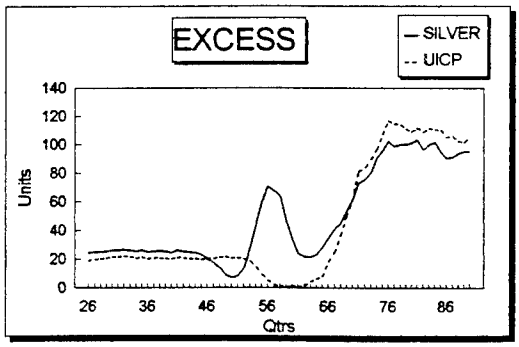






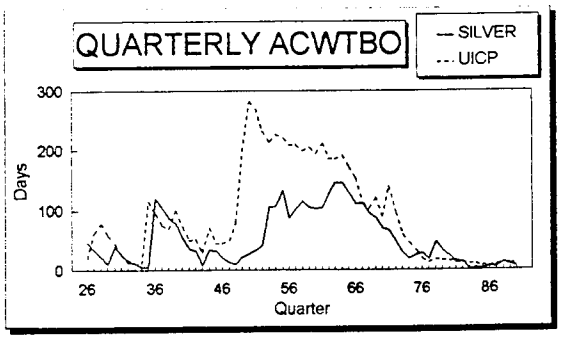


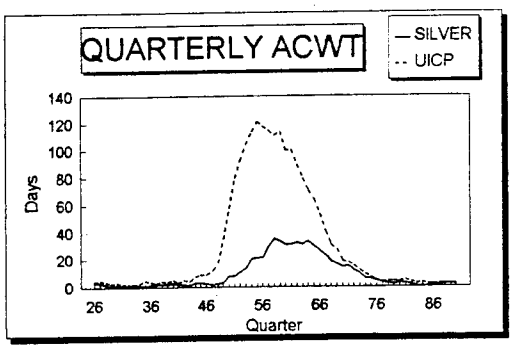


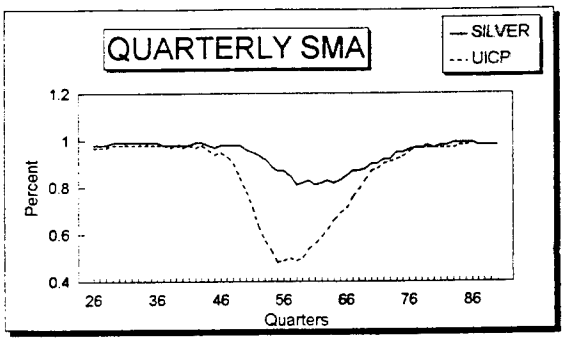


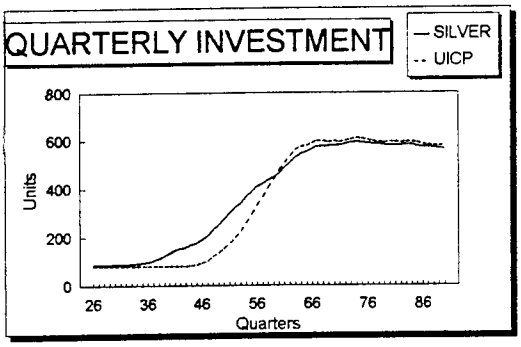
APPENDIX K. INCREASING DEMAND (SLOW TREND) GRAPHS

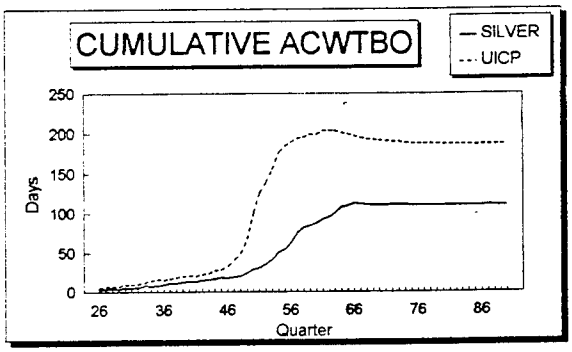
Description: Increasing Experiment # 79

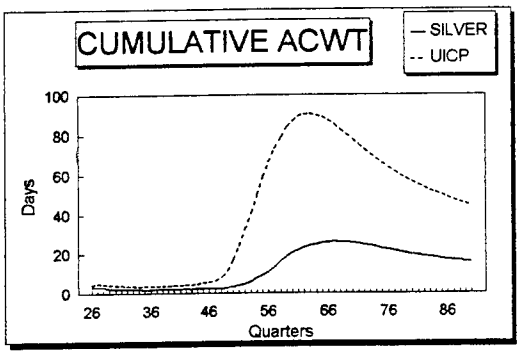


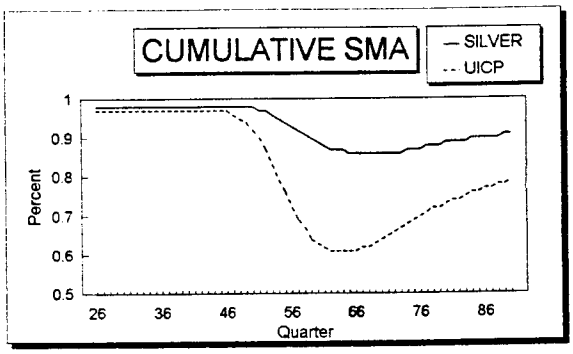


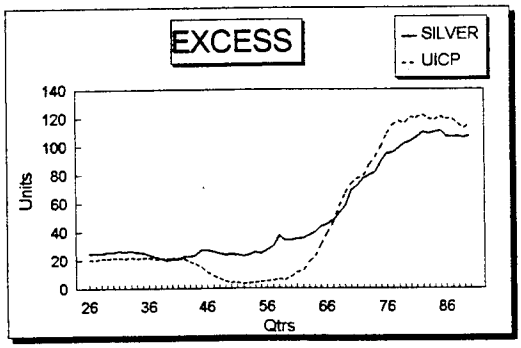












LIST OF REFERENCES

1. United States General Accounting Office/National Security and International Affairs Division Report #92-112, "Cost Factors Used to Manage Secondary Items," Washington, D.C., May 1992.

2. United States General Accounting Office/High Risk Report #93-12, "Defense Inventory Management," Washington, D.C., December 1992.

3. United States General Accounting Office, Draft Report, "Applying Commercial Purchasing Practices Should Help Reduce Supply Costs," Washington, D.C., Draft April 1993.

4. Lilli, C.M., and Husson, C.R., "Wholesale Level Reorder Point and Reorder Quantity Computation During Periods of Declining Demand," Master's Thesis, Naval Postgraduate School, Monterey, California, December 1992.

5. Perry, J.H., "Growth in Unneeded Inventories: Contributing Factors," Logistics Spectrum, Vol. 25, No. 2, p. 19-25, Summer 1991.

6. United States General Accounting Office, National Security and International Affairs Division, Report # 91-176, "Shortcomings in Requirements Determination Processes," Washington, D.C., May 1991.

7. Silver, E., "Replenishment Under a Probabilistic Time-Varying, Demand Pattern," AIIE Transactions, Vol. 10, No. 4, p. 371-379, December 1978.

8. Wagner, H.M.,and Whitin,T.M. "Dynamic Version of the Economic Lot Size Model," Management Science, Vol. 5, No. 1, October 1958.

9. Silver, E., and Peterson, R., Decision Systems for Inventory Management and Production Planning, Second Edition, John Wiley and Sons, Inc., 1985.

10. Blackburn, J.D., and Millen, R.A., "Heuristic Lot-Sizing Performance in a Rolling-Schedule Environment," Decision Sciences, Vol, 11, No. 4, p. 691-701, October 1980.

11. Ritchie, E., and Tsado, A.K., "A Review of Lot-Sizing Techniques for Deterministic Time-Varying Demand," Production and Inventory Management, Third Quarter, p. 65-79, June 1986.

12. Cline, B.S., Foote, B.L., and Schlegel, R.E., "Multicriteria Evaluation of Lot Sizing Techniques as a Function of Demand Pattern, Time Between Orders, and Demand Variability," USAF Technical Report 91-5, April 1991.

13. Donaldson, W.A., "Inventory Replenishment Policy for a Linear Trend in Demand - An Analytical Solution," Operational Research Quarterly, Vol. 28, p. 663-670, July 1977.

14. Ritchie, E., and Tsado, A., "The Penalties of Using the EOQ: A Comparison of Lot Sizing Rules for Linear Increasing Demand," Production and Inventory Management, First Quarter, p. 12-17, January 1986.

15. Askin, R.G., "A Procedure for Production Lot Sizing with Probabilistic Dynamic Demand," AIIE Transactions, Vol. 13, No. 2, p. 132-137, June 1981.

16. Bollapragada, S., and Morton, T.E., "The Non-Stationary (s, S) Inventory Problem Near-Myopic Heuristics, Computational Testing," Working Paper # 1993-01, Graduate School of Industrial Administration, Carnegie Mellon University, February 1993.

17. Nanda M. Balwally, Defence Electronics Supply Center to Dr. Thomas Moore, Naval Postgraduate School, letter dated 14 January 1993.

18. Department of Defense Instruction 4140.39, "Procurement Cycles and Safety Levels of Supply for Secondary Items," July 1970.

19. Hadley, G., and Whitin, T.M. Analysis of Inventory Systems, Prentice-Hall, Inc., 1963.

20. OPNAV Instruction 4440.23, "Procurement Cycles and Safety Levels of Supply for Secondary Items," February 1976.

21. U.S. Department of the Navy, Supply Systems Command, NAVSUP Publication 553, Inventory Management, January 1991.

22. Functional Description - PD82, "Uniform Inventory Control Program - Levels Setting Model," FMSO Document Number FD-PD82, April 1993.

23. Interview between J. Boyarski, Navy Ship's Parts Control Center, Code 041, Mechanicsburg, PA and the author, 26-29 May 1993.

24. Bissinger, B.H., and Boyarski, J.R., "A Rank Correlation Approach for Trend Detection of Military Spare Parts Demand Data," Thirty-Sixth Conference on Design Experiments, October 1990.

25. Brown, R.G., Smoothing, Forecasting and Prediction of Discrete Time Series, Prentice-Hall, Inc., 1963.

26. Ross, S.M., Introduction to Probability Models, Fourth Edition, Academic Press, Inc., 1989.

27. Bickel, P.J., and Doksum, K.A., Mathematical Statistics: Basic Ideas and Selected Topics, Holden-Day, Inc., 1977.

28. Zehna, P.W., "Forecasting Errors Using MAD", Naval Postgraduate School, Technical Report NPS55Ze9014A, April 1969.

29. Law, A.M., and Kelton, W.D., Simulation Modeling and Analysis, Second Edition, McGraw-Hill, Inc., 1991.

30. Cormen, T.H., Leiserson, C.E. and Rivest, R.L., Introduction to Algorithms, The MIT Press and McGraw-Hill Book Co., 1991.

31. Bunker, T., "Demand Forecasting Simulation," Program Documentation, Navy Ships Parts Control Center, 1987.

32. Mendenhall, W., Wackerly, D.D., and Scheaffer, R.L., Mathematical Statistics with Applications, Fourth Edition, PWS-Kent Publishing Co., 1990.

33. Interview between K. Reynolds, Navy Ships Parts Control Center, Code 046, Mechanicsburg, PA, and the author, 26-29 May 1993.

34. Hwang, C., and Yoon, K., Multiple Attributes Decision Making - Methods and Applications, Lecture Notes in Economics and Mathematical Systems, Vol. 186, Springer-Verlag, 1981.

35. Tersine, R.J., Principles of Inventory and Materials Management, Third Edition, Elsevier Science Publishing Co., 1988.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 052 Naval Postgraduate School Monterey, California 93943-5002	2
3.	Defense Logistics Studies Information Exchange United States Army Logistics Management Center Fort Lee, Virginia 23801-6043	1
4.	Professor Thomas P. Moore, Code SM/Mr Department of Systems Management Naval Postgraduate School Monterey, California 93943-5103	1
5.	Professor Alan W. McMasters, Code SM/Mg Department of Systems Management Naval Postgraduate School Monterey, California 93943-5103	1
6.	Professor Gordon H. Bradley, Code OR/Bz Department of Operations Research Naval Postgraduate School Monterey, California 93943-5103	1
7.	CDR Eduardo DeGuia, Code 4111 Naval Supply Systems Command Washington, D.C. 20376-5000	1
8.	Mr. Michael Pouy HQ-Defense Logistics Agency [ATTN: MMSB] Cameron Station Alexandria, Virginia 22304-6100	1

9. Mr. Jere Engleman, Code 046 — 1
Navy Ships Parts Control Center
5450 Carlisle Pike
P.O. Box 2020
Mechanicsburg, Pennsylvania 17055-0788

9. Mr. John R. Boyarski, Code 0412 — 1
Navy Ships Parts Control Center
5450 Carlisle Pike
P.O. Box 2020
Mechanicsburg, Pennsylvania 17055-0788

10. Mr. Tom Lanagan — 1
Headquarters, DLA
ATTN: DORO-Supply Analysis
c/o: Defense General Supply Center
Richmond, Virginia 23297-5082

11. Mr. Alan Kaplan — 1
Army Material Systems Analysis Activity
800 Custom House
Second and Chestnut Street
Philadelphia, Pennsylvania 19106

12. LT Glenn C. Robillard — 1
6 Foxmeadow Drive
Worcester, Massachusetts 01602

13. Mrs. Fran Gabriel — 1
Aviation Supply Office
700 Robbins Avenue
Philadelphia, Pennsylvania 1911-5098

14. LCDR Kevin Maher, Code 041 — 1
Navy Ships Parts Control Center
5450 Carlisle Pike
P.O. Box 2020
Mechanicsburg, Pennsylvania 17055-0788